UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Development of a mobile application for vertical jump detection

MASTER'S DEGREE FINAL WORK

Master's Degree in Computer Engineering

Author: Ismael Pérez Martín

Tutor: M. Carmen Juan Lizandra

Course 2020-2021

Abstract

This Master's Degree final work consists of the development of an Android application to detect a vertical jump through a video and show its data and measurements. The system is intended to be used by people who want to record or improve their vertical jumping ability or otherwise want to evaluate it from a health point of view.

This purpose arises from the idea of improving current applications to analyze vertical jump. The idea for improvement comes from the fact that none of the existing applications, even the market leaders, present the possibility of detecting the jump automatically. Thanks to this, the idea arose of creating an application that allows the user to obtain the measurements of their vertical jump in a more automatic way, without having to view a complete video to inform the application when the jump is made.

The first idea of this work is to carry out a research work on the tools and methods to detect a vertical jump in a video. This investigation was carried out in this order: a phase of analysis of the different tools for their detection, another phase of analysis of the ways to detect the jump with the chosen tool, a phase of tests to verify that the pre-established precision requirements are reached, and finally, an analysis of the results to establish some guidelines for the performance of the vertical jump.

The results obtained from the research are that first, a vertical jump is detectable in a video by analyzing contours and their movements in the Y axis of the frames, the best position to detect it in a video is to stand sideways to the camera, and finally, the best distance to stand from the camera to detect the jump is 120cm.

As the final result obtained, the user is offered a system to add videos of vertical jumps through an Android application and that they are processed automatically without their intervention. This processing is responsible for obtaining all the respective measurements to the jump and showing them to the user in a simple way. Finally, a system is provided to the user to view their history of validated jumps together with the measurements and relevant aspects of each of the vertical jumps.

Key words: vertical jump, computer vision, Android, OpenCV, automatic detection

Contents

List of Figures

List of Tables

CHAPTER 1
Introduction

Jump monitoring devices are often high-cost devices that operate by pressure or laser detection. To solve this problem, several applications for vertical jump analysis emerged, which are positioned as market leaders but not with very good ratings or results. The worst point of this kind of applications is the need for a manual selection of the takeoff and landing moments of the jump in the video.

From this point arises the idea of offering the users a jump analysis option through automatic take-off and landing detection. This idea provides a fundamental value to the users since they only have to be in charge of recording their jumps with certain guidelines and they will directly obtain the information in the application (height, time in the air, power, etc).

Vertical jumping power is becoming increasingly important in terms of health and sports performance, which is why it is oriented both to a market of athletes who do not have the resources to afford professional equipment and to people who want to improve their health by practicing jumps.

The most critical aspect of the project is to provide a good detection of the two critical moments of the jump: takeoff and landing. This is the main differentiating aspect with the competition so it must have an appropriate accuracy and performance that make the product distinctive and attractive to the user.

This work covers the development of a vertical jump detection system that meets the needs of the market and serves as a minimum viable product to evaluate if the idea can be the basis of a business idea. This development is carried out in two main parts: A preliminary investigation to evaluate the possibility of automatic detection, and if it meets the necessary requirements, and a part of implementation of the idea to be able to translate the minimum viable product.

1.1 Motivation

This chapter details and justifies the different aspects related to the project carried out. These aspects encompass both personal motivations that justify aspects such as technologies or programming languages, up to the interest in possible career opportunities. All these reasons for carrying out this Master's final project are detailed below:

- Analyze and try to improve an existing idea.
  The main aspect to achieve success with a product that people always think about is to create a new idea, but with this project exists the possibility of analyzing a

product created in advance, finding its weaknesses and strengths and trying to create a product of better quality, offering something that may be of further benefit to the user.

- Create a minimum viable product for a business idea.
  As it is about analyzing and improving an existing product, this idea is the basis for evaluating whether this idea is valid for a possible commercial application or not.

- Learn computer vision technologies.
  Due to the improvement of these technologies such as OpenCV [1], their use is much greater in the industry, opening both new research branches and new professional opportunities, since there are even masters dedicated exclusively to these technologies.

- Learn mobile application development.
  Due to the high use of mobile devices in society, mobile applications are completely necessary in order to offer the user a satisfactory experience when using a service. This field also has many career opportunities and is a highly valued career option.

- Improve the knowledge of Python.
  This programming language is one of the most useful and well-known, but during studies it has been used in a very limited way in small projects. Using this language on a larger scale as a master's final project allows a more in-depth look at its capabilities and possible uses.

- Learn how to create a scalable microservices framework from scratch.
  By including the project with both backend and frontend, and having more weight on the work in the backend, it is an opportunity to learn how to create scalable microservices to be able to divide that workload.

- Apply the knowledge acquired about statistics.
  Because this work requires an initial research phase, the results must be analyzed using statistical methods and resources. This adds a greater value to the work since it does not cover only the field of software development.

- Put into practice the knowledge learned.
  Being a complex project without initial details, it is the opportunity to use the different knowledge acquired during the studies to analyze, design and create a system from its beginning to end.

The motivations detailed above explain the aspects that make this project of such interest, establishing a starting point for the development of the objectives detailed below.

1.2 Objectives

The main objective of this work is to carry out an investigation about vertical jump detection methods, in order to carry out an implementation of a minimum viable product of an application that automatically detects vertical jumps, offering the data to the user. In order to meet this main objective, the following sub-objectives must be met:

- Analyze the available computer vision tools by comparing their strengths and weaknesses.

[1] OpenCV official webpage: https://opencv.org/

- Choose and learn about the most suitable computer vision tool for the project.
- Research about the best methods to detect a vertical jump in a video.
- Perform tests with the different methods, configurations and videos.
- Analyze the different detection methods results to obtain the one with the highest precision and stability.
- Implement the chosen detection method as a microservice accessible by REST API.
- Develop the backend through REST API to provide the application with the necessary methods for the detection of the vertical jump and use of the database.
- Configure the different environments to host the detection microservice, the backend, the database and the storage of the videos.
- Deploy the different services replicated in a scalable way.
- Design and implement an Android application that uses the services deployed.
- Carry out real tests with the whole system deployed.

These objectives are responsible for guiding the entire process that will be carried out during the development of the application. They are the basis for specifying the result to be achieved and establish the starting point of the work to be done.

1.3 Methodology

The realization of this project comes from an already established idea with clear objectives and with little flexibility, so a final version is expected and not several versions with greater functionality. The project is more focused on offering a limited functionality with high reliability, so an iterative development is not the best option. In addition to this, the project has very different phases that depend on the previous one, so it must be developed in an established order without skipping any phase and with a low possibility of repetition of any phase.

Another point to keep in mind is that, because it has a research part, the development depends on the final solution chosen and its implementation. Finally, for this project there is only one person in charge of its development, so tasks or stages of its development cannot be paralleled.

With all these reasons, the conclusion is reached that the best methodology to follow for its development is a waterfall methodology [2]. This methodology allows to initially specify the final objectives and, following the phases in an orderly and consecutive manner, to arrive at a final product that meets these requirements.

Waterfall development is a linear procedure that is mainly characterized by dividing the development processes into successive phases of the project, as shown in figure 1.1. Unlike iterative models, each phase of the project runs only once. The results of each of the phases serve as a starting point for the next phase.

This model is widely used, especially for projects for public entities or large corporations that seek a final result with very detailed specifications. The methodology

[2] Waterfall methodology deep specification: https://learn.marsdd.com/article/product-development-the-waterfall-methodology-model-in-software-development/

is often highly criticized due to its high cost in modifications. This is because these modifications are usually required when seeing the final result of the process, in which all the phases have already been passed, and it is necessary to go through each of them again in order to implement or modify the required functionality. This can be due to both customer requirements and errors found in an advanced phase of the project.

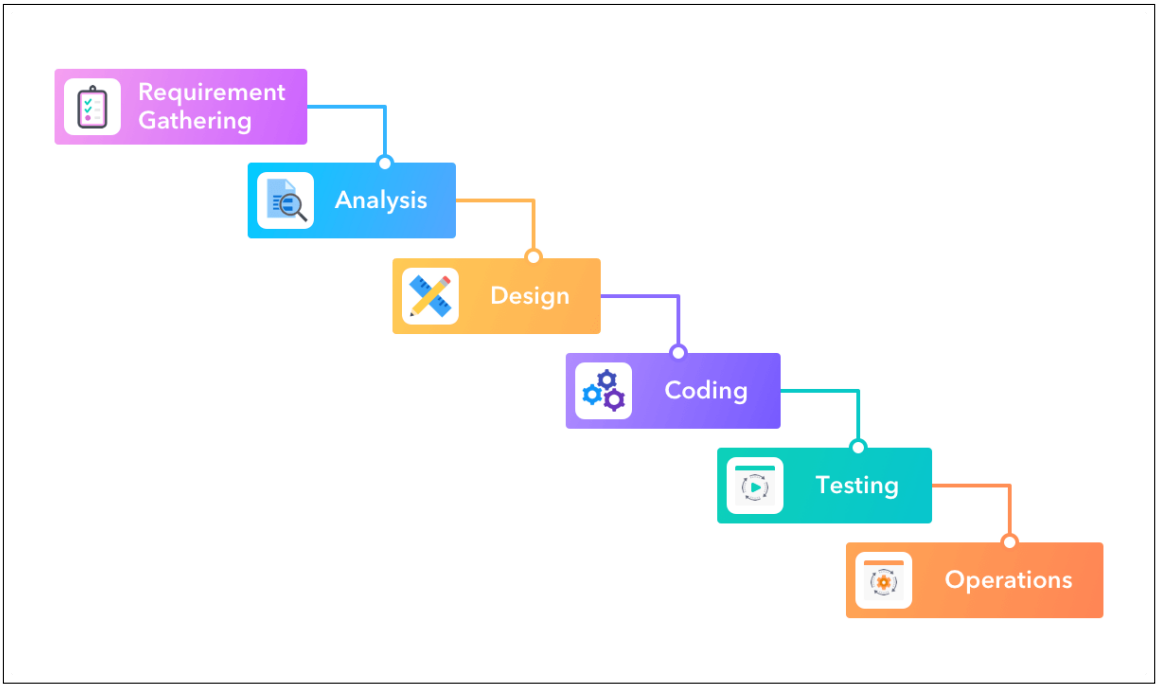


Figure 1.1: Waterfall methodology phases schema

In the development of this project, a final validation of a client is not necessary, which reduces the risk of modifying functionalities, although it does not reduce the risk of finding possible errors at an advanced stage. Having the requirements defined from a project base also reduces this possibility, since the work has a final idea to present and everything is focused on that result from the analysis phase.

After this phase, the design and implementation phase should be carried out, but since this work also consists of a research part, this is slightly modified. Research on methods and forms of detection of vertical jump requires tests to analyze their results, so after the analysis phase there is a joint investigation, development and testing stage. Thanks to doing this in a parallel and more or less cyclical way, it allows to automate test processes to later evaluate new forms or detection methods in the future. This phase is lengthened until a certain percentage of error is achieved, so its estimation is quite difficult due to the lack of knowledge in this field, since there are no similar studies.

After completing the investigation by reaching the goal error percentage, the consecutive phases of development and testing continue, mixing both phases in an intermediate stage for a higher quality of the product, thus being able to resolve the errors in a faster and more effective way.

1.4 Memory structure

This document consists of the following chapters:

1. **Introduction, motivation and objectives**: This chapter places a global context in which the work is to be carried out, as well as the reason for its development and its ultimate goal.
2. **State of the art**: This chapter captures the current technological situation regarding the idea to be developed, analyzing the options already available in the market and critiquing them.
3. **Proposal**: This chapter reflects the analysis of the system requirements together with their use cases and conceptual models to establish which is the knowledge and technological space of the project.
4. **Proposed solution**: This chapter shows the proposed solution, its development phases, the details of its architecture and how the system will be implemented and validated.
5. **Implementation and tests**: this chapter presents the system implementation stage, the tests carried out and the results obtained from the tests.
6. **Conclusions**: It is the last chapter dedicated to analyzing whether the objectives have been achieved and presenting problems that may have arisen during development, as well as presenting possible future work.

CHAPTER 2
State of the art

The development of mobile technologies has allowed a great advance in communication technologies and, in turn, an increase in the appearance and use of social networks, causing a large part of the population to be registered on this type of platform [1], as can be seen in the figure 2.1, with a very big impact on young people.

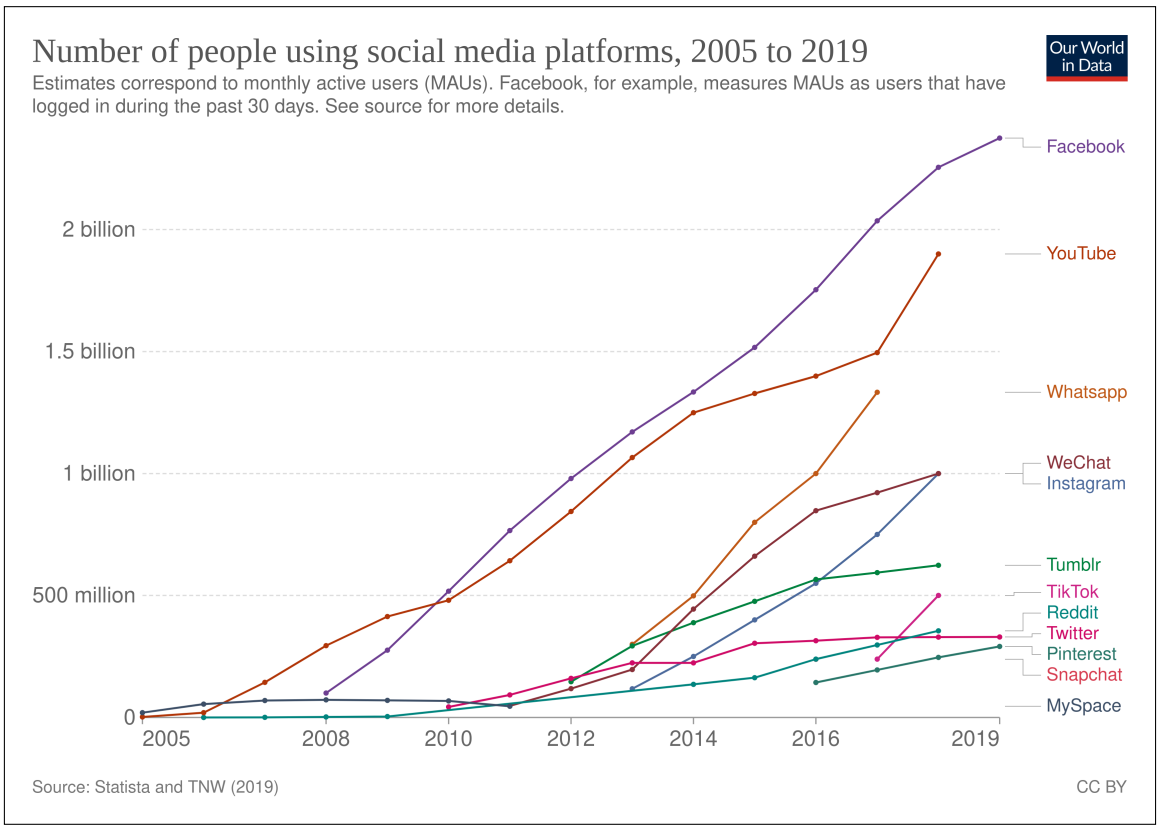


Figure 2.1: Number of people using social media platforms, 2005 to 2019

This factor has caused that, in recent years, the images of the so-called "influencer" of social networks have become popular. These influencers also have a great impact in the world of sports, since there is a very large aspect that are fitness influencers. They show through social networks their physique and training methods, showing how they have obtained their results and motivating other people to achieve it. These people usually

[1] The rise of social media: https://ourworldindata.org/rise-of-social-media

have an athletically developed physique, from elite athletes to bodybuilders, which establishes a physical aspect that the rest of the population tries to achieve.

Along with these data, another important aspect to take into account in the increase in the popularity of sport in Spain. The figure 2.2 shows the data collected in the Sports Habits Surveys of the Ministry of Culture and Sport of Spain [1] [2].

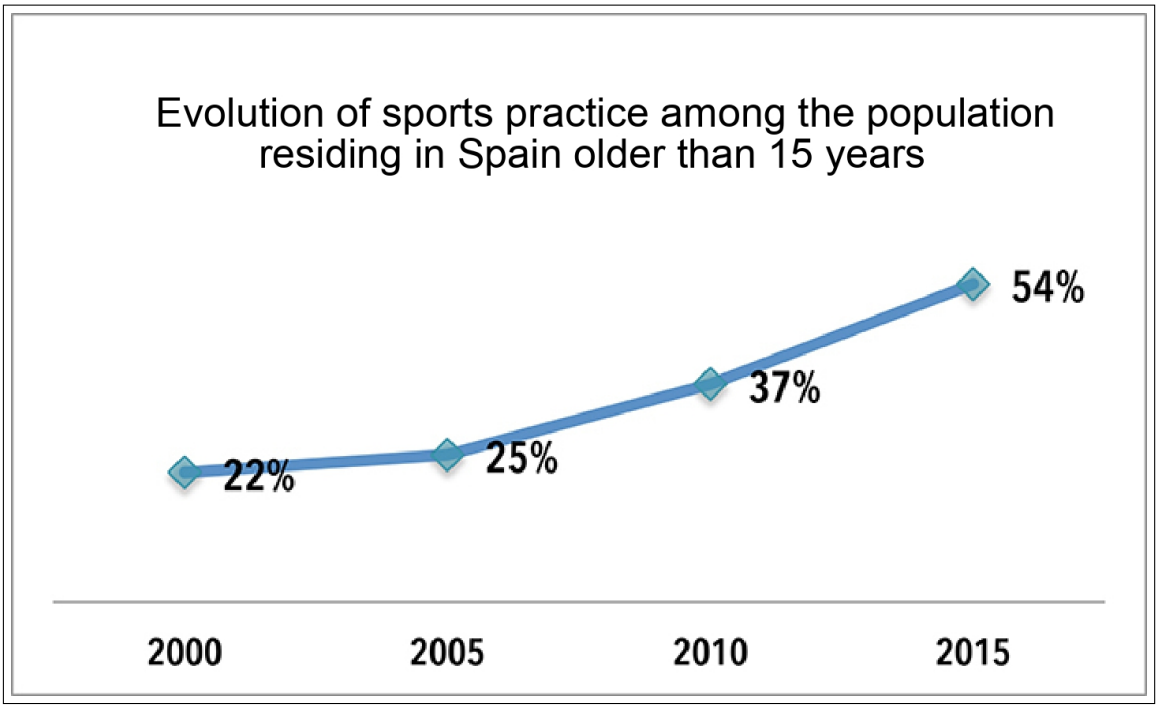


Figure 2.2: Evolution of sports in the spanish population older than 15 years

If we contrast these data together with the graph of the evolution of social networks, we observe that the increase in the practice of sport also occurs together with the increase in the use of social networks. These data also make sense if it is observed that the main reason for practicing sports is to be in shape, to look better physically, as shown in the figure 2.3 obtained through the data from the aforementioned surveys.

Another aspect that social networks have expanded is the popularity of the feats of some elite athletes, such as Cristiano Ronaldo's header [2]. Such is the repercussion that it has even become a game as can be seen in figure 2.4. Elite athletes like the mentioned Cristiano Ronaldo or Anthony Jerome "Spud" Webb winning a dunk contest being only 1.68m tall [3], show the public that a good vertical jump is related to being fit. This relationship is demonstrated in studies such as the one carried out by Darmiento et al. [3], which shows a correlation of vertical jump power with athletic performance, or the one carried out by Kinet [4], which shows the correlation of vertical jump power with squat power.

[2]Cristiano Ronaldo header video, slow motion analysis at 1:10 second: https://www.youtube.com/watch?v=AunImole9HI

[3]Spud Webb wins 1986 NBA Slam Dunk Contest: https://www.youtube.com/watch?v=r1YRJvFvlgg

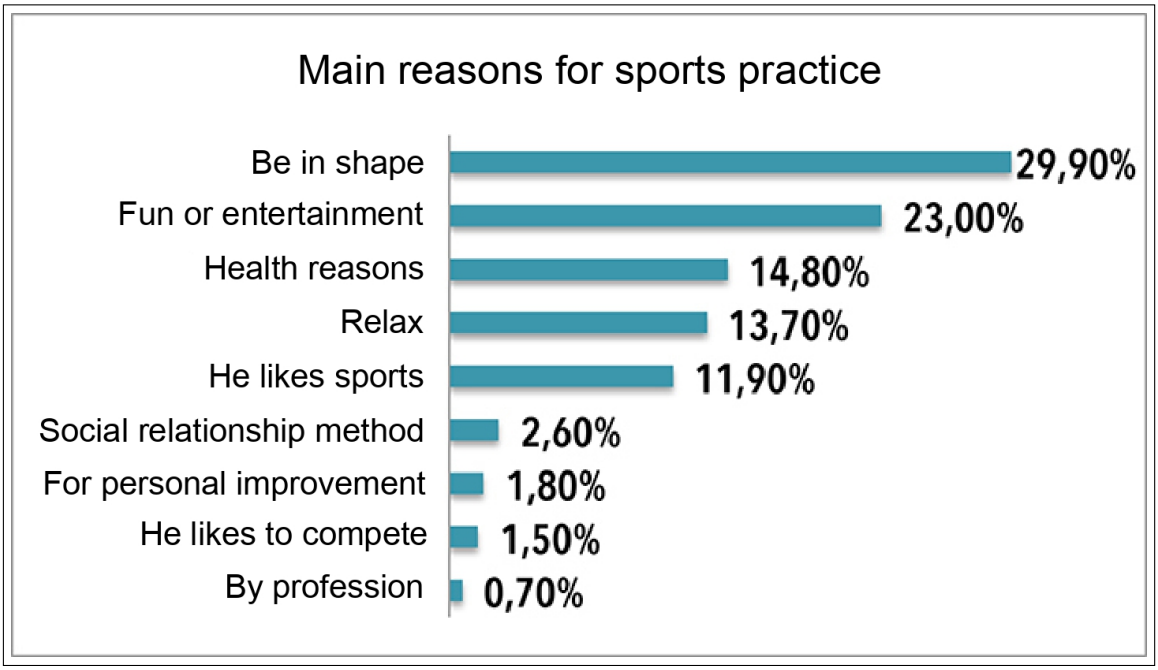


Figure 2.3: Main reasons for practicing sports in the Spanish resident population

Figure 2.4: Cristiano Ronaldo header challenge

All these events and developments in the sport have also led to an increase in vertical jump detection methods. This has evolved from more rudimentary methods such as marking the height reached on a wall with the hand to laser and pressure devices for the detection of the jump, which are collected in studies such as those carried out by Moir [5] or like the webpage of Wood [4]. These methods range from low price and moderate

[4]How to measure vertical jump: https://www.topendsports.com/testing/products/vertical-jump/index.htm

accuracy to thousands of dollar equipment such as laser detection or using two highly sensitive pressure plates.

With the evolution of sport and mobile devices, this leads to the creation of solutions to measure vertical jump on mobile devices. These solutions use aspects such as sensors of the phone like the microphone as the final degree project carried out at this university to obtain the height of the vertical jump through sound processing [6]. All these solutions are detailed below, presenting their strengths and weaknesses with further analysis.

2.1 Existing solutions

In a very global way for this project, we can differentiate the existing solutions for the detection and vertical jump analysis into two large blocks:

2.1.1. Solutions using physical tools

These are the most common and most used over time, ranging from simple push mechanisms to complicated laser or pressure sensing systems. The methods discussed in this subsection present a classification in terms of how to measure vertical jump.

Measuring Distance

These devices are based on devices placed on the wall that the user himself uses to mark the height manually. All of these methods have two phases:

1. Raise the arm in a static position to obtain the base height from which the jump is to be made.
2. Mark manually on the device the height reached jumping.

We can highlight the following methods:

- **Wall mounted** [5]
  A scale located on the wall in which the user marks with his hand how high he has jumped.

Figure 2.5: Wall mounted vertical jump measurement method

[5]Details of wall mounted measurement method: https://www.topendsports.com/testing/products/vertical-jump/wall-mounted.htm

- **Vertec** [6]
  Consisting on horizontal vanes which are rotated out of the way by the hand to indicate the height reached.

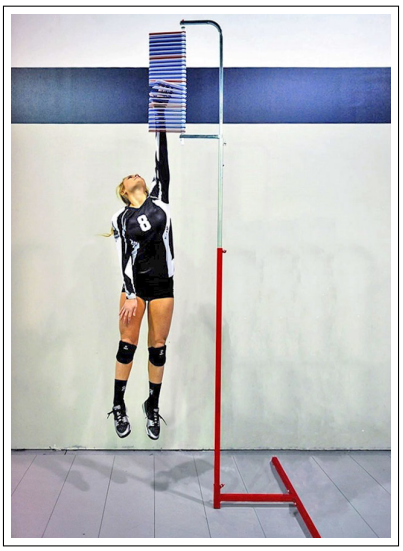


Figure 2.6: Vertec vertical jump measurement method

- **Vertical jump mat** [7]
  Consisting of a tape tied to a belt by means of which, after setting the measurement to zero before the jump, when jumping the tape will mark the height reached.

Figure 2.7: Mat vertical jump measurement method

[6] Details of Vertec measurement device: https://www.topendsports.com/testing/products/vertical-jump/vertec.htm

[7] Details of mat measurement device: https://www.topendsports.com/testing/products/vertical-jump/jumpmat.htm

Ground sensor contact devices

These devices are based on sensors placed at the base of the jump by which they detect the moment of jump and fall through changes in the pressure applied to the platform. The two main stages detected are:

1. Takeoff
   This is detected by the application and subsequent lack of pressure on the platform.

2. Landing
   This is detected by applying pressure again after the jump moment.

We can highlight the following methods:

- **Just Jump System** [8]
  Consisting on a mat with hand-held battery that calculates vertical jump height by measuring the time that the feet are not in contact with the mat. The jump height appears directly on the plate display.

Figure 2.8: Just Jump vertical jump measurement method

- **Force plate** [9]
  Consisting on detect the jump height by the pressure exerted on one or two metal plates. The jump results are displayed on a system connected to the plates.

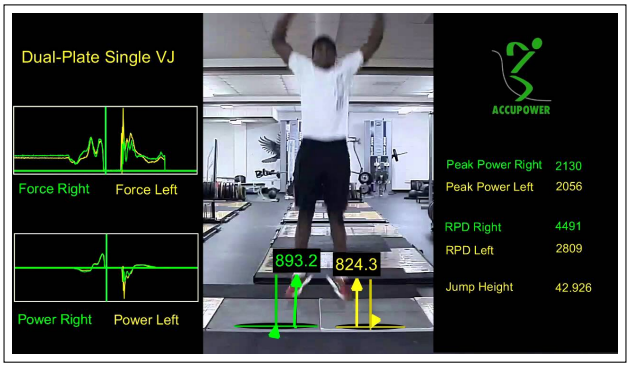


Figure 2.9: Force plate vertical jump measurement method

[8] Just Jump System link with details: https://www.performbetter.com/Just-Jump-System_2
[9] Details of what is a force plate: https://www.hawkindynamics.com/blog/what-is-a-force-plate

- **ChronoJump** [10]
  It consists of a computer tool for measurement, management and statistics of short-term sports tests. It can be used to measure the jump in height thanks to its pressure detectors.

Figure 2.10: ChronoJump measurement kit

Photoelectric Circuits (Laser/Infrared)

These devices base their operation on the detection of cuts in the light by laser or infrared, thus detecting with these cuts in the light flow the movements of the user when performing the high jump. We can highlight the following:

- **Optoelectronic device** [11]
  Consisting of a ground sensor for measuring the time the feet leave the floor, and another sensor mounted at around waist height to measure the height reached in the vertical jump.

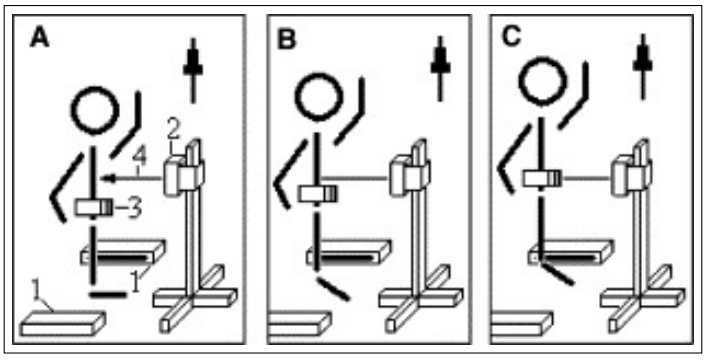


Figure 2.11: Optoelectronic vertical jump measurement method - Take off

[10] ChronoJump official webpage: https://chronojump.org
[11] Details of optoelectronic devices: https://www.topendsports.com/testing/products/vertical-jump/optoelectronic.htm

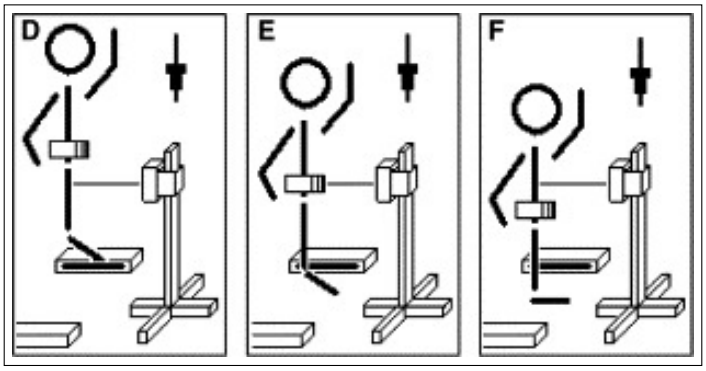


Figure 2.12: Optoelectronic vertical jump measurement method - Landing

- **Ground-Based Laser/Infrared Beams** [12]
  Consisting of a beam of light located at the base of the jump. Timing starts when the light at ground level becomes unbroken, and finishes when the subject lands and breaks the beam. With these data, the rest of the jump data is calculated. An example of the use of this technology is for example G-Flight system [13]. The jump results are shown directly on the device display.

Figure 2.13: Ground-Based Laser/Infrared Beams vertical jump measurement method

- **Brower measurement** [14]
  Its installation is the same as the Wall mounted but in this case it consists of a touchless device that automatically detects the height reached by hand as in Wall mounted, but more precise.

[12] Details of laser detection devices: https://www.topendsports.com/testing/products/vertical-jump/laser.htm
[13] G-Flight official webpage: https://strongbyscience.net/2017/12/01/g-flight/
[14] Details about Brower device: https://www.topendsports.com/testing/products/vertical-jump/brower.htm

Figure 2.14: Brower vertical jump measurement method

Accelerometer

An accelerometer is called any instrument designed to measure accelerations. Using this sensor, some wearables are able to measure the height and power of the jump with the help of the gyroscope on some occasions. The most notable options on the market are:

- **Vert** [15]
  This wearable placed on the user's waist or in a garment adapted for the device. This device measures by means of the accelerometer and the gyroscope the height reached in the vertical jump.

Figure 2.15: Vert vertical jump device

[15] Vert official webpage: https://www.myvert.com/?lang=es

- **Accelerometer based applications**
  As in the previous case, these applications measure the height of the vertical jump using the accelerometer and gyroscope of the mobile device itself. An example of this is Jumpster [16], which through the accelerometer obtains the jump data while keeping the device in your pocket.

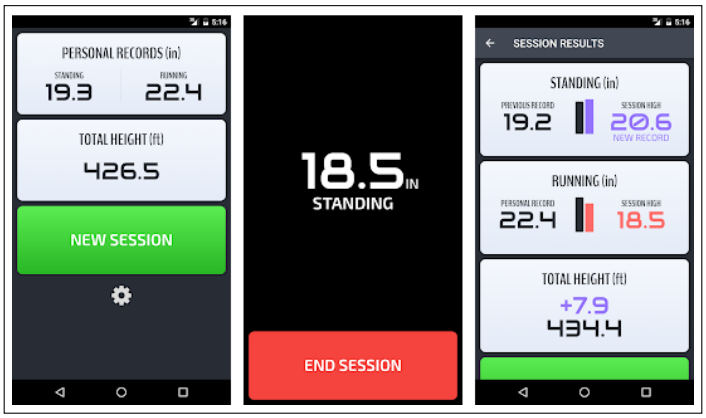


Figure 2.16: Jumpster app

2.1.2. Solutions using video

These solutions are based on, by means of only a video of the vertical jump, being able to extract the necessary data for the user, without any type of equipment as in the previous cases. These options are based on analyzing a jump video by timing or using the device's own distance measurement tools that records the vertical jump. The two most prominent options are the height measurement from video and the timing using video.

Height measurement from video

This methods consist of using the device's measurement tools or calculating the distance analyzing a video to extract the height manually.

- **Using the device's height measurement tools**
  Consisting of, using the device's measurement tools, analyzing a video to extract the height reached by the user at a specific moment in the video, the highest point. An example of this is the distance meter that iPhones have on their camera[17]. Using the frame where the greatest height is reached, this tool can be used to measure the distance from the ground to the jumper's feet.

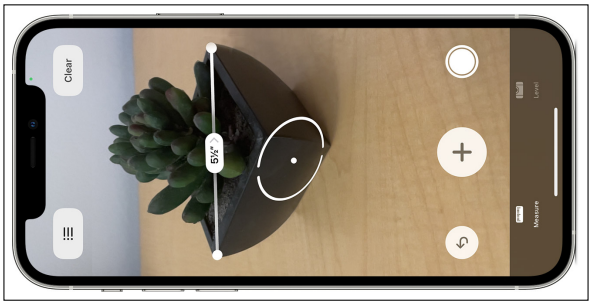


Figure 2.17: iPhone camera measure tool

[16] Jumpster official webpage: https://skyhawkmedia.com/jumpster.html
[17] iPhone measurement tool details: https://support.apple.com/en-us/HT208924

- **Manually analyzing jump height**
  This method consists of manually calculating the distance from the camera to the jump and once the jump is made, analyzing the height reached with the video and the established parameters.

Figure 2.18: Vertical jump height measurement method from video

Timming using video

This is the option used by most of the mobile apps available for vertical jump height measurement. It is based on, once the vertical jump has been recorded on video, manually choosing the takeoff and landing moment of the jump. After having this data, the application calculates the power and height of the jump using the time in the air. The most relevant applications on the market that use this technique are listed below:

- **MyJump 2** [18]
  This is the leading application in both the Android and iOS market to measure vertical jump through timing. Its interface allows the user to choose both moments of the jump in a fairly intuitive way. After choosing the jump takeoff and landing in the video, calculate the rest of the vertical jump parameters from the time in the air. Its appeal is found in the amount of data provides to the user about his jumps. Its creator Carlos Balsalobre [19] has several sports applications apart from MyJump 2 such as Runmatic or Nordics.

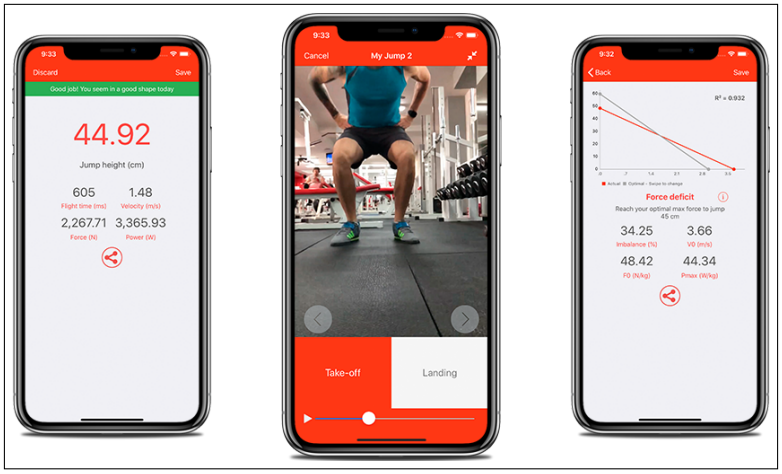


Figure 2.19: MyJump 2 app

[18] MyJump 2 app iOS https://apps.apple.com/es/app/my-jump-2/id1148617550 and Android https://play.google.com/store/apps/details?id=com.my.jump&hl=es&gl=US official websites
[19] Carlos Balsalobre official webpage: https://www.carlos-balsalobre.com/index.html

- **Whats My Vertical?** [20]
  This application for iOS is based on the same operation but with a slightly less updated interface and with some different functions. It uses the same operation as the previous application, after choosing the jump takeoff and landing in the video, it calculates the rest of the vertical jump parameters from the time in the air. Its appeal is found in functions such as calculating how much you have to jump to make a mate based on your height and setting that goal.

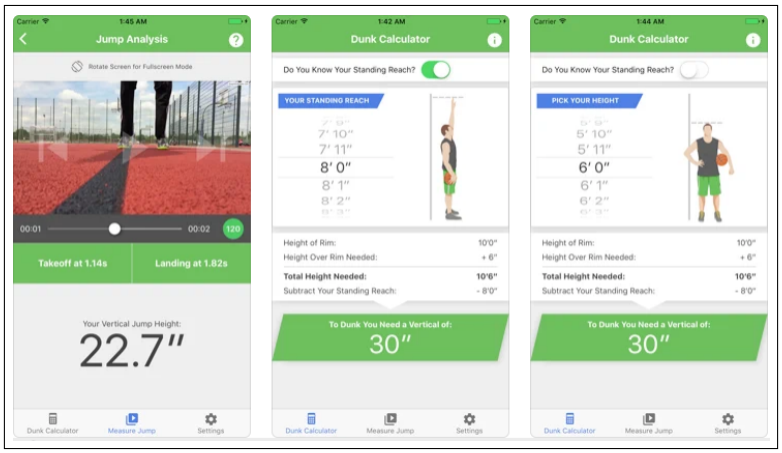


Figure 2.20: Whats My Vertical? app

2.2 Critical analysis of the existing solutions

Because the solution to be developed is based on a mobile application, all solutions that use physical devices, except those based on accelerometer detailed on the subsection 2.1.1, are discarded for analysis. The main aspect that makes it discarded is that the solution to be developed seeks to be easy to use by any user at any time, without having to take into account having a tool at hand to be able to measure the jump or spend large sums of money as in the case of laser or pressure devices.

Another aspect to rule out these options is the existence of studies that validate the precision of application-based solutions, such as the one carried out by Montalvo et al. [7], in which it is concluded that they are a very good option to comparison of high cost devices. Next, the analysis of the existing solutions is carried out, which use a mobile device for detection as in this project.

2.2.1. Accelerometer

These methods are detailed in the 2.1.1 subsection. The accuracy of accelerometer-based methods is high [8], which provides a strong point for these types of solutions.

- **Vert**
  Its price of around 175$ is not the highest but it is not cheap for an average user, so it is a negative point for this solution. Its ease of use is good as only one sensor needs to be placed on the waist, but it must be well positioned to obtain good application results. As for its precision, it is based on the precision of the device and is high. The accelerometer analysis provides a fairly good level of detail.

[20] Whats My Vertical? app official website: https://apps.apple.com/us/app/whats-my-vertical/id1237413241

Finally, in terms of the data presented and its quantity, they are sufficient and well presented in their application. Its user interface is intuitive and up-to-date for the user.

- **Accelerometer based applications**
  This application is free for both Android and iOS, so it is a plus. As for its use, it is enough to have the phone in your pocket and jump, so it is good in terms of the devices to use and their use. Its accuracy is good but can be affected by the movement of the phone in your pocket and it may vary from phone to phone by its accelerometer. One drawback could be its user interface, which is outdated in some examples like Jumpster and can be a negative point for the user. As for its data, it is sufficient and well presented but its user interface is outdated and is not entirely intuitive for the user.

2.2.2. Height measurement from video

These methods are detailed in the 2.1.2 subsection.

- **Using the device's height measurement tools**
  If you have a device that has these tools, the price is free but the device is required. Only the mobile device and a video are needed, so it is a point in favor of the devices to be used, but one more step is necessary analyzing the highest frame of the jump, so its difficulty of use increases. As for its precision, again it resides in the precision of the tool and it can vary depending on the background that the video frame has. As for the data, only the height is obtained and the user must calculate the time so it is neither sufficient nor well presented. The interface on the other hand is intuitive since it is to select the ground and the highest point of the jump.

- **Manually analyzing jump height**
  The price is a strong point since it is only necessary a fixed mobile device, a clear bottom and calculate the distance to the jump from the phone. This in turn is good since you don't need a lot of devices. On the other hand, its difficulty of use increases due to having to perform manual calculations to obtain the height. Its precision may be low due to bad calculations by the user of the distances between the mobile or some effect of the camera. As for the data provided, it is very poor and its presentation is null since the user must calculate it, so these are negative points of this solution.

2.2.3. Timing using video

These methods are detailed in the 2.1.2 subsection.

- **MyJump 2**
  The cost of this application is low but not zero, around 15$, so it is not one of its advantages. It stands out for only having to use a mobile device, but its use is not the simplest either, since the user manually chooses the moment of takeoff and landing from the vertical jump. As for its precision, it is based on the precision of the user when choosing both moments of the jump. The phone also has an influence since if you record at more fps the analysis of the moment will be easier. If both moments of the jump have been chosen well, its precision is high since it has been contrasted with pressure plate results in several studies [9] [10]. Finally, its amount

of data and presentation are good since it uses a user interface with good usability and intuitive, so the results of the jump are easily seen.

- **Whats My Vertical?**
  This application is quite similar to the previous one but differs in that, first, its price is considerably lower, almost zero, which is an advantage, but its user interface is more obsolete. The usability is similar but it is less up-to-date despite having some greater functionality than the previous one like the dunk calculator.

2.3 Computer vision available technologies

This section will detail the video processing tools currently available on the market. Each of them will highlight their strengths that can be useful for this solution. The most prominent technologies currently are detailed in the following subsections.

2.3.1. Amazon Recognition

Amazon Rekognition [21] is capable of processing both images and videos, detecting in them faces, objects, movements, actions and even things like inappropriate content. It is a very powerful tool that offers the user a wide menu of options, which makes it very usable by almost any user.

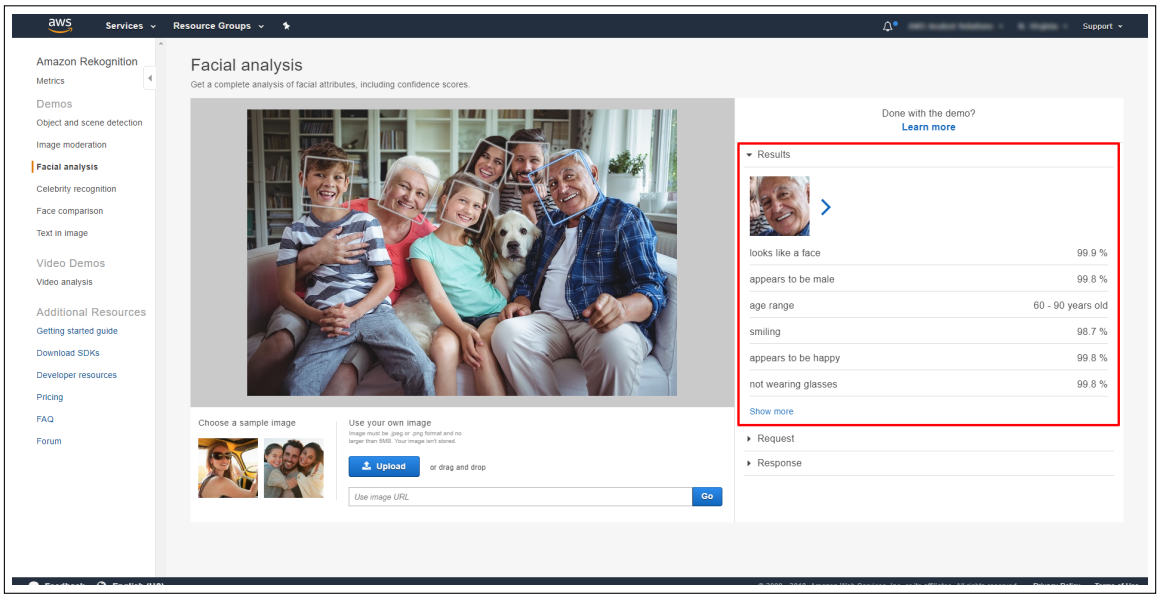


Figure 2.21: Amazon Rekognition tool

Advantages

- **Great support**
  Amazon provides its users with great support regarding the use of its tools, so it is easy to contact Amazon support for any problem or doubt.

- **Easy to use**
  The user interface is very usable and its menus are dedicated to users who are not too expert, so anyone can use it.

[21] AmazonRekognition official website: https://aws.amazon.com/rekognition/

- **Very good facial recognition**
  Their facial recognition is a very prominent aspect in their opinions.

Disadvantages

- **Expensive**
  The price of this tool is high, so an investment must be made for its use.

- **Slower to reach ROI**
  This is due in part to its price, but the Return On Investment is slower than other technologies to which it competes.

- **Its high usability limits functionalities**
  Because it is aimed at all types of user, some more technical options are limited and cannot be used by more expert users.

2.3.2. Azure Video Analyzer

Azure Video Analyzer [22], formerly known as Microsoft Video API, is a cloud-based API with capabilities like tracking faces, detecting motion, or stabilizing a video. It enables the creation of more personalized and intelligent applications by automatically understanding and transforming video content.

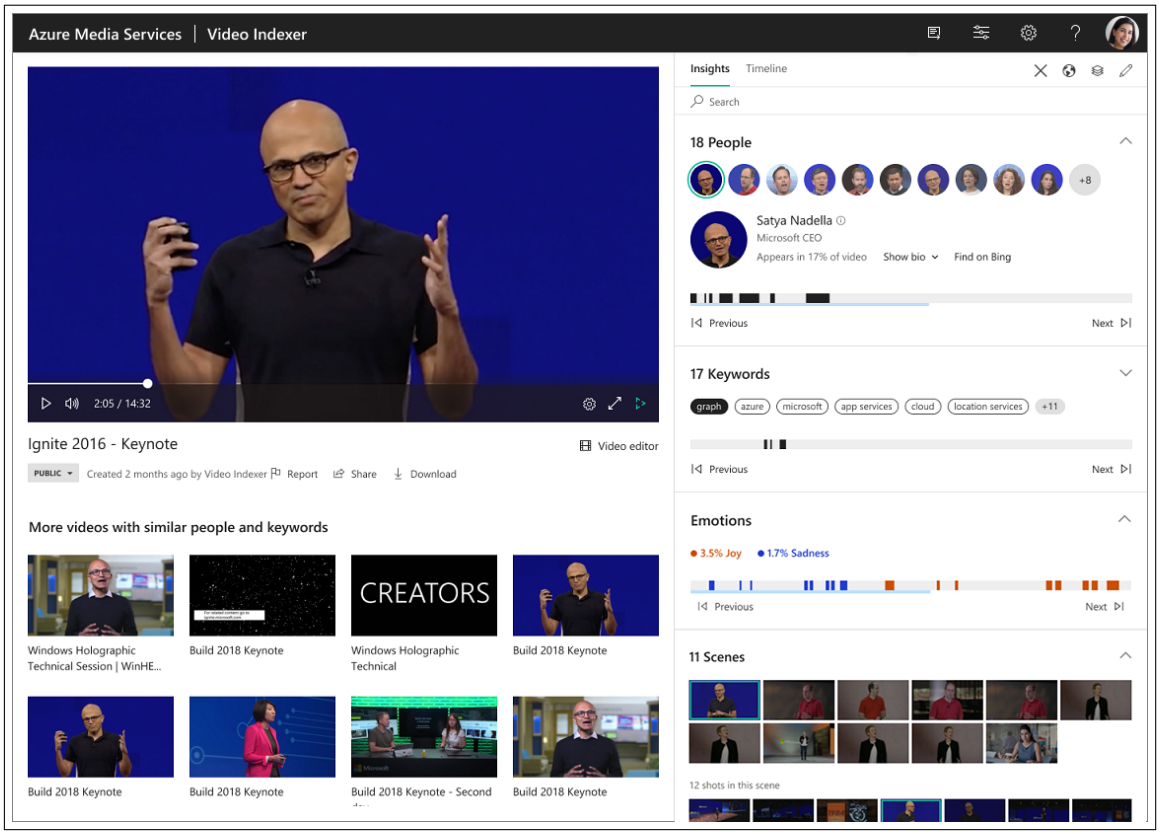


Figure 2.22: Azure Video Analyzer tool

[22] Azure Video Analyzer official website: https://azure.microsoft.com/en-us/products/video-analyzer/

Advantages

- **Better at meeting requirements**
  This tool offers some more specific functionalities that offer greater power to the tool.
- **Easy usable API**
  The way its API is implemented makes its use very easy and intuitive for the user.

Disadvantages

On the other hand, about its disadvantages it is important to highlight the following:

- **Expensive**
  The price of this tool is high, so an investment must be made for its use.
- **Slower to reach ROI**
  This is due in part to its price, but the Return On Investment is slower than other technologies to which it competes.
- **Some bugs**
  It is a tool still under development as azure itself indicates in the "preview" tag. This means that it still has some bugs.

2.3.3. OpenCV

OpenCV is a library available for C, C++, Python, and Java and is compatible with Windows, Linux, Mac OS, iOS, and Android for image and video processing with a strong focus on real-time applications. It can take advantage of multi-core processing and is enabled to take advantage of the hardware acceleration of the underlying heterogeneous computing platform. It is one of the leading tools in terms of image and video processing.

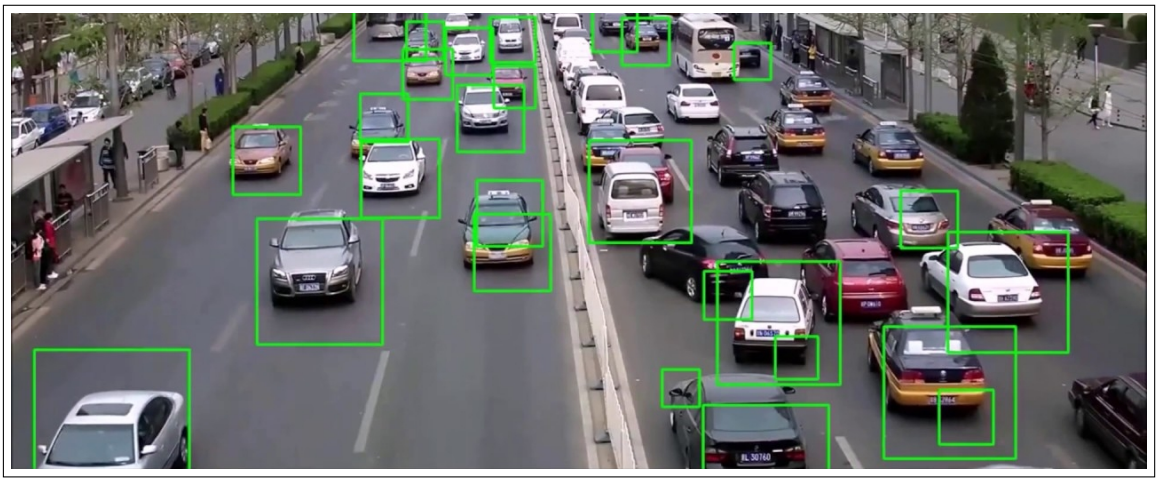


Figure 2.23: OpenCV cars detection example

Advantages

Regarding its advantages, we can highlight the following:

- **Open Source**
  This means that its price is zero and that everyone can contribute to its continuous development.
- **Multiple Interfaces**
  This tool is available for several programming languages and platforms, being equally fast in all of them since its original C / C ++ code is executed.
- **Programming Functions Library**
  Its library offers a great multitude of functions to use all its capabilities, resulting in an infinite combination of possibilities of use.
- **Community**
  Being one of the most famous image and video processing tools, it has a large community behind it with many examples and help in its forums.

Disadvantages

On the other hand, about its disadvantages it is important to highlight the following:

- **Sometimes single-core use if not using C/C++**
  Some of its functions work in parallel but others cannot be configured unless a language with multiprocessor management is used.
- **Takes more time to show the processed result**
  As for its competitors, this tool takes more time to show the processing done to the image or video

2.3.4. SimpleCV

SimpleCV [23] is an open source framework for creating computer vision applications, the user can access various high-powered computer vision libraries such as OpenCV without having to learn about all the technical aspects necessary to use such libraries.

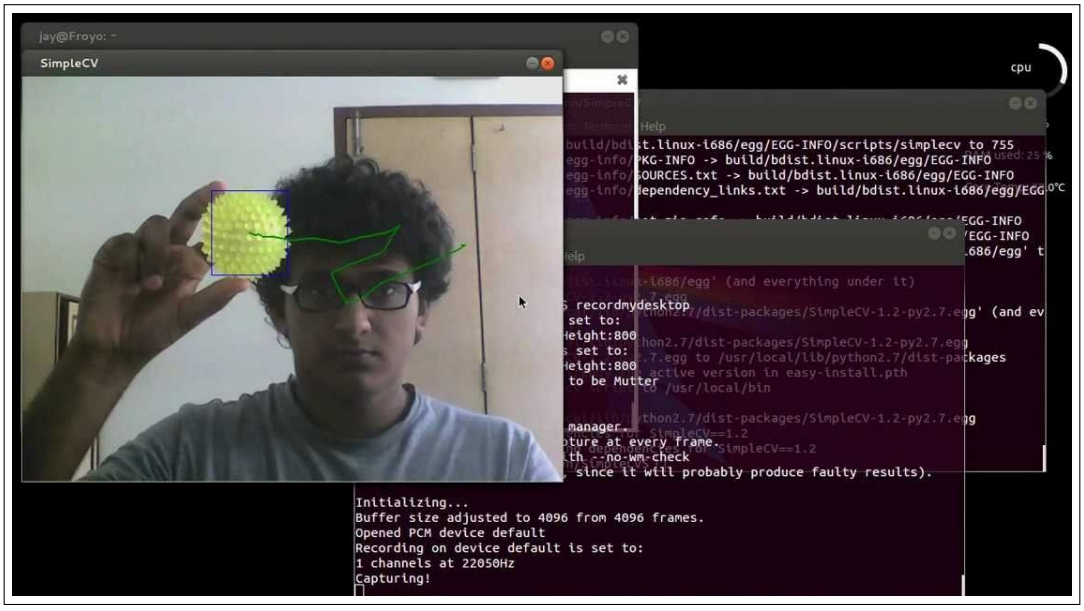


Figure 2.24: SimpleCV ball tracking example

[23] SimpleCV official webpage: http://simplecv.org/

Advantages

- **Open Source**
  This means that its price is zero and that everyone can contribute to its continuous development.
- **Usability**
  This tool is dedicated to simplifying the technical aspects of frameworks such as OpenCV, so its usability increases for users without that technical knowledge.
- **Good support**
  This technology leads to greater support for the frameworks it uses, its own support, which provides added value.
- **Good documentation**
  Its documentation is very good and detailed, which helps to be able to use all its capabilities

Disadvantages

- **Its high usability limits functionalities**
  Because it is aimed at all types of user, some more technical options are limited and cannot be used by more expert users.
- **Only available for Python**
  Unlike some of the frameworks on which it is based, it is only available for this programming language, which forces you to use Python to use this tool.

2.3.5. TensorFlow

TensorFlow, owned by Google, is one of the open-source machine learning platforms with a great set of tools and libraries, available for various languages such as Python, C ++, Java or Javascript. It is another of the leading tools in terms of image and video processing [24]. This technology is used mainly to develop computer vision applications using machine learning. It allows to train models in a very simple way to be able to detect from objects to movements.

Advantages

- **Open Source**
  This means that its price is zero and that everyone can contribute to its continuous development.
- **Multiple Interfaces**
  This tool is available for several programming languages, which makes it possible to transfer examples made in one language to another.
- **Programming Functions Library**
  Its library offers a great multitude of functions to use all its capabilities, resulting in an infinite combination of possibilities of use.

[24] TensorFlow object detection example: https://www.tensorflow.org/hub/tutorials/object_detection

- **Community**
  Being one of the most famous image and video processing tools, it has a large community behind it with many examples and help in its forums.

Disadvantages

- **Consumes a very high amount of resources**
  This library is normally used together with CUDA to take advantage of parallel computing capabilities, so the amount of resources it consumes rises very quickly.
- **Needs a previous training of the model to be used**
  The model to recognize movements or objects needs a previous training of the model, and that implies the need to have a large set of examples for the model, something that is not available in the case of this project.

2.4 Technological context

This chapter analyzes the technologies to be used in the project, explaining their advantages and disadvantages, the use they will have within the system and a comparison with technologies similar to the one chosen.

2.4.1. OpenCV

The details of this open source computer vision library have been detailed previously in the subsection 2.3.3. This library has been chosen to implement the processing and detection of the vertical jump, detecting both its takeoff and landing moments. The main reasons why it has been chosen over the rest of the options are:

1. **Availability for various programming languages**
   My knowledge of Python is not extremely advanced, so being able to also find examples in languages like Java or C in which I have more knowledge can be of great help in expanding the base knowledge for developing the project.
2. **Community**
   The wide community of this library makes examples available for many types of situations and uses of the library, also making it possible to receive help from other users with knowledge of OpenCV.
3. **Open Source**
   This project is carried out without any external financing, so avoiding the costs of large platforms such as Amazon or Microsoft is a point to consider.
4. **Large number of functions in its library**
   This makes it possible that several different configurations can be tested to obtain the one that best detects the vertical jump.

2.4.2. Python

Python is an interpreted programming language whose philosophy emphasizes the readability of its code. It is a multi-paradigm programming language, since it partially supports object-orientation, imperative programming and, to a lesser extent, functional

programming. This programming language is used both for the development of the vertical jump detection tool in a video, and for the development of the application's backend. The main reasons why this programming language has been chosen over other options are:

1. **Previous knowledge**
   Despite not being the programming language that I master the most, my knowledge of this language is extensive and I feel comfortable developing with it.

2. **Community**
   Being the most famous programming language right now [25], its community is one of the largest. This provides added value when it comes to finding solutions to specific language problems.

3. **Available libraries**
   This language has an extensive repository of libraries that can be useful for any of the steps in the development of the solution.

2.4.3. Kotlin

Kotlin is a statically typed programming language that runs on top of the Java virtual machine and can also be compiled into JavaScript source code. It is a language oriented to object-oriented programming, with the purpose of improving Java [26] but being interoperable with this language. This programming language is used for Android application design, as it is one of the official Android programming languages. The main reasons for choosing this programming language are:

1. **Similarity to Java**
   Due to the previous knowledge of Java, adapting to this language is very easy, in addition, it offers the possibility of using Java code if some part of it is not possible with Kotlin due to lack of knowledge.

2. **Less code required than in Java**
   Since its purpose is to improve Java, many of Java's tedious operations such as constructors are already in place to save development time.

3. **Popularity in the job market**
   Due to its popularity as a language, its popularity in terms of job offers has also risen, so learning this language can open up future job opportunities.

2.4.4. Flask

Flask is a minimalist open source framework written in Python that allows you to create web applications quickly and with a minimum number of lines of code [27]. This framework is used to create encapsulation of the backend web applications and the vertical jump detector developed in Python, accessed through the REST API. Its implementation allows creating web applications using an MVC pattern with very few lines, having a large number of libraries to complement its operation. Its authentication and cookie management is very simple and it also has a unit test system.

[25] PYPL GitHub: https://pypl.github.io/PYPL.html
[26] Advantages of Kotlin over Java: https://alignminds.com/advantages-kotlin-over-java/
[27] How to create a Flask app: https://flask.palletsprojects.com/en/2.0.x/quickstart/

2.4.5. MongoDB

MongoDB [28] is one of the most popular non-relational database systems today. Its most distinctive aspect is that, unlike relational databases, it stores data as Json documents in key-value pairs. The main advantage of these databases is their flexibility to modify the predefined schema. This allows new fields to be added or modified during development without having to use a new database schema.

Another aspect to highlight is its scalability, since MongoDB parallelizes the serial queries, it is very easy to create several instances that store the data in the same place. This system is used to create and modify the database through its Python client, Pymongo [29], thus allowing to maintain the consistency of the application as can be seen in the figure 2.25.

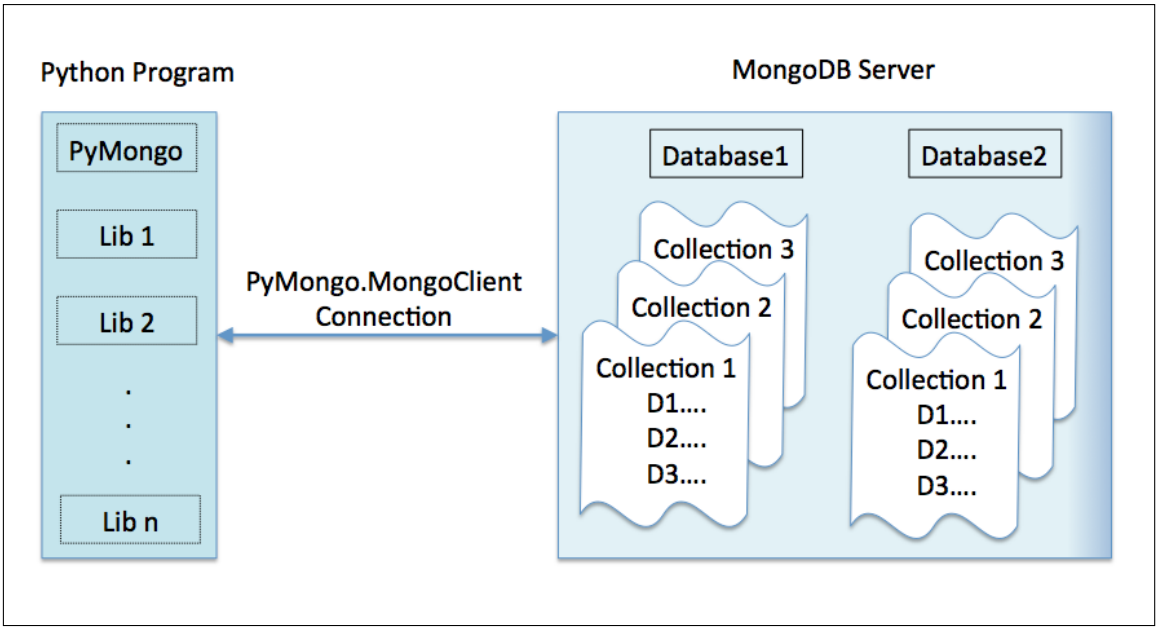


Figure 2.25: Pymongo query schema

2.4.6. Git

Git is version control software designed by Linus Torvalds, with the efficiency, reliability and compatibility of application versioning in mind when they have a large number of source code files. Its main characteristics are:

- Focused on a non-linear development, being able to create branches (versions) of the project.
- The management is distributed, each programmer has his copy in his local repository.
- Repositories can be published using protocols such as HTTP, SSH or TCP / IP.

[28] What is MongoDB and how it works: https://www.mongodb.com/en/what-is-mongodb
[29] Pymongo use tutorial: https://pymongo.readthedocs.io/en/stable/tutorial.html

This version control system is used to manage all parts of the product [30], having a repository for each of them: vertijump-detector, vertijump-backend, vertijump-db and vertijump-app. The creation of branches for their proper development is done following the GitFlow methodology [31], explained in figure 2.26.

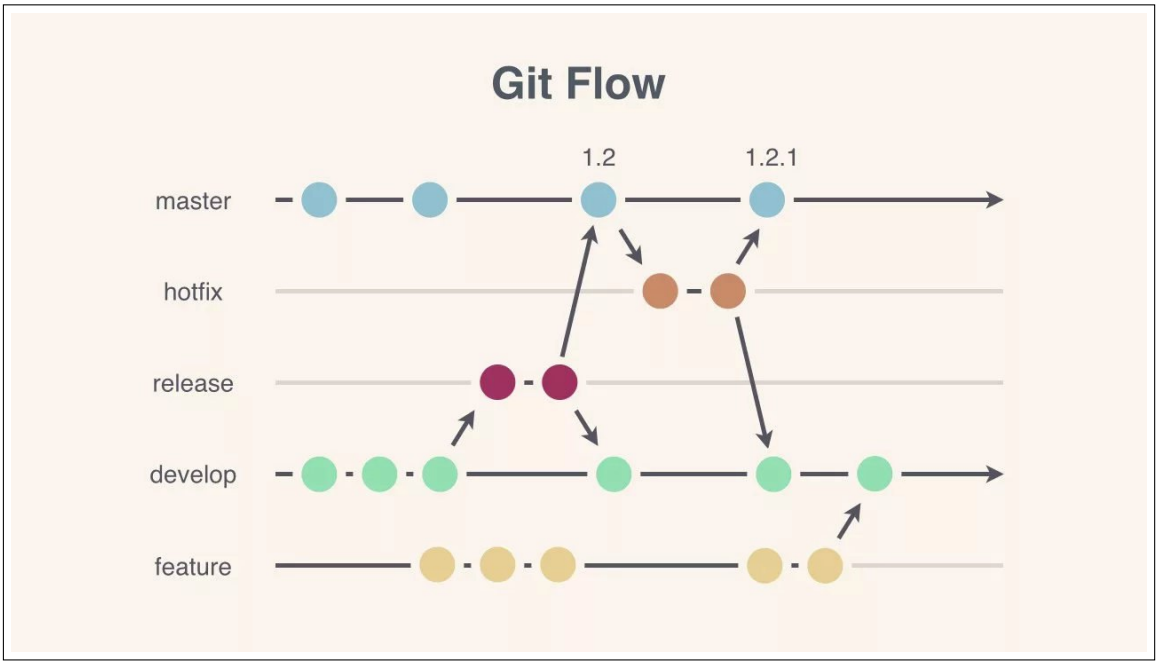


Figure 2.26: GitFlow schema

Importantly, Git is version control, but GitHub [32] is used to host the repositories named above. This web platform is the most used for managing repositories and includes very interesting aspects such as CI/CD [33], management of pull requests with code reviews or its own markdown language.

2.4.7. Docker

Docker is an open source project that automates the deployment of applications within software containers, providing an additional layer of abstraction and application virtualization automation across multiple operating systems. This tool is used to encapsulate each part of the project [34], thus facilitating the possibility of deploying the service in an autonomous, fast and secure way, since the deployment environment is always the same. The images are created from a Dockerfile in which all the container parameters are specified [35].

To make full use of this tool, Docker Compose is used [36], a Docker tool for running multi-container Docker applications. This is done through an .yml file, in which the

[30] Why use a version control: https://www.git-tower.com/learn/git/ebook/en/desktop-gui/basics/why-use-version-control/
[31] GitFlow workflow: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow
[32] Why use GitHub: https://apiumhub.com/tech-blog-barcelona/using-github/
[33] GitHub CI/CD systems: https://docs.github.com/es/actions/guides/about-continuous-integration
[34] Benefits of containerization: https://hentsu.com/docker-containers-top-7-benefits/
[35] Create Docker images with Dockerfile: https://docs.docker.com/engine/reference/builder/
[36] Why use Docker Compose https://www.enterprisedb.com/postgres-tutorials/why-you-should-use-docker-compose

services to be deployed are detailed and the tool allows us to deploy all of them in a single command. This tool is highly used in CI / CD systems, by which automatic Docker deployments can be scheduled when certain conditions are met. Another important aspect is the ability to scale applications, allowing you to run multiple instances of the same container and manage them through a load balancer.

Finally, in order to store these Docker images, the official Docker registry for image management DockerHub is used as can be seen on the figure 2.27 . It offers possibilities such as version control or tag of specific versions [37]. In this registry the images of all the created services are stored, together with the images of their load balancers for their later deployment in a scalable way.

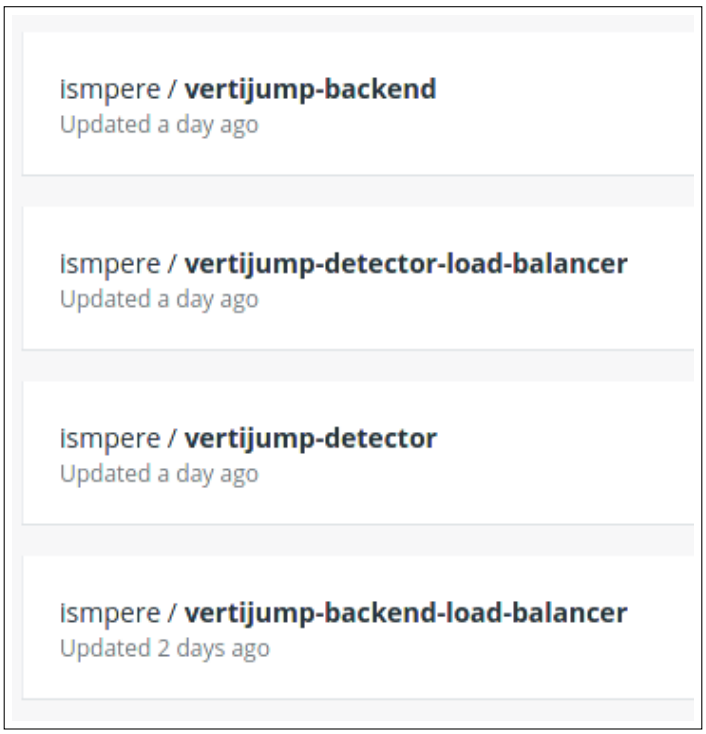


Figure 2.27: DockerHub project images

[37] Docker repositories management: https://docs.docker.com/docker-hub/repos/

CHAPTER 3
Proposal

The proposal for this project is to develop a minimum viable product of an automatic vertical jump detection application. This application is called Vertijump.

As detailed in the chapter State of the art, there are several mobile applications designed to obtain the data from a vertical jump. Some of these applications like the detailed in the section 2.1.1 make use of external sensors such as the accelerometer, having to carry the device during the jump, a factor that can affect the result if the device is not properly fixed. These applications have a fast and quite precise operation, as it has been possible to verify in several investigations mentioned previously, for which they are not an objective to improve.

The other great part of applications like MyJump 2 or Whats My Vertical? detailed in the section 2.1.2, are based on video timing. The user is in charge of, once the vertical jump has been recorded, manually selecting in which frame of the video the takeoff is made and in which the landing is made [1]. After selecting these moments, they obtain the time in the air of the vertical jump by calculating the difference between both moments. The rest of the parameters are calculated in a fast way, since they are simple calculations for a mobile device to have the time in the air.

Although the obtaining of results is fast, the part of selecting the moments of the jump is somewhat tedious for the user, since it is based on observing a video frame by frame, a process that can be very slow the more frames per second the video has. Another negative factor in the case of MyJump 2 is the need to sometimes manually insert the frames per second of the video [2], an aspect that an average user is likely not to know.

The aim of the proposal is to solve this problem, automatically detect the takeoff moment and the landing moment of the vertical jump together with the frames per second of the video, thus obtaining the user the data of the vertical jump automatically without selecting the jump moments manually. Instead of viewing frame by frame, the user will only have to trim the video by choosing the vertical jump part and all the jump data will be obtained automatically when processing the video.

The rest of the functionalities are similar to the rest of the applications since the main objective is to obtain that differentiating aspect compared to the rest of the competitors. This application, being a minimum viable product, can serve as the basis for a future evaluation of the viability of a company based on this product.

[1] Tutorial MyJump 2: https://www.youtube.com/watch?v=TnLT3IREioE

[2] Video of MyJump 2 insert frames per second requirement: https://www.youtube.com/watch?v=nvsU2EtwPPI&t=204s

3.1 Definition of the actors

The actors are those involved in the interaction with the system. In order to continue with the analysis of the system, it is necessary to define the actors that will use it. The main actors are listed below.

3.1.1. Unidentified user

This actor has only the basic capabilities to perform the actions pertinent to registration and login in the system.

3.1.2. Identified user

This actor, being already registered in the system, can use its full capabilities after logging into the system in addition to the capabilities of the unregistered user.

3.2 Analysis of requirements

The requirements for a system are the description of what the system must do, the service or services it provides, and the limitations on its operation.

The specification of the requirements must be precise and unambiguous since they describe what the system must do in specific situations, its limitations and specifications to meet quality requirements. These requirements are obtained through a requirements elicitation techniques [11], which allows analyzing the needs of the system in an appropriate way. After applying these techniques, the requirements obtained for the system are detailed below.

3.2.1. Functional requirements

A functional requirement defines a function from a software system or its components. A function is described as a set of inputs, behaviors, and outputs. These kinds of requirements are the specific functionalities that a system is supposed to fulfill. The functional requirements obtained for the system are detailed below.

Unregistered user

Code	RF-01
Name	Sign up
Dependencies	
Description	The application must allow the user to register in the system by email, password and the remaining attributes of the user.
Importance	High
Restrictions	

Table 3.1: Functional requirement RF-01 Sign up

Code	RF-02
Name	Login
Dependencies	RF-01
Description	The application must be accessible through an authenticated access by email and password.
Importance	High
Restrictions	

Table 3.2: Functional requirement RF-02 Login

Registered user

Code	RF-03
Name	Visualize main menu
Dependencies	RF-02
Description	The application must allow the logged in user to visualize the main menu.
Importance	High
Restrictions	

Table 3.3: Functional requirement RF-03 Visualize main menu

Code	RF-04
Name	Visualize vertical jumps
Dependencies	RF-03
Description	The application must allow the user to view a list of his vertical jumps.
Importance	Medium
Restrictions	

Table 3.4: Functional requirement RF-04 Visualize vertical jumps

Code	RF-05
Name	Visualize vertical jump
Dependencies	RF-04
Description	The system must allow the user to see the details of a vertical jump from his list of vertical jumps.
Importance	Medium
Restrictions	

Table 3.5: Functional requirement RF-05 Visualize vertical jump

Code	RF-06
Name	Add vertical jump
Dependencies	RF-03
Description	The system must allow the user to add a new vertical jump.
Importance	High
Restrictions	

Table 3.6: Functional requirement RF-06 Add vertical jump

Code	RF-07
Name	Add video from gallery
Dependencies	RF-06
Description	The system should allow the user to add a video from the gallery.
Importance	High
Restrictions	

Table 3.7: Functional requirement RF-07 Add video from gallery

Code	RF-08
Name	Trim video
Dependencies	RF-07
Description	The system should allow the user to cut a selected video.
Importance	High
Restrictions	

Table 3.8: Functional requirement RF-08 Trim video

Code	RF-09
Name	Detect vertical jump
Dependencies	RF-08
Description	The system must allow the user to detect a vertical jump from a video.
Importance	High
Restrictions	

Table 3.9: Functional requirement RF-09 Detect vertical jump

Code	RF-10
Name	Validate vertical jump
Dependencies	RF-09
Description	The system must allow the user to validate the results of a vertical jump.
Importance	High
Restrictions	

Table 3.10: Functional requirement RF-10 Validate vertical jump

3.2.2. Non-functional requirements

Non-functional requirements specify criteria that can be used to judge the operation of a system rather than its specific behaviors. These requirements can be related to issues such as efficiency, response speed, load supported or aspects such as technologies to use. The non-functional requirements extracted from the system are detailed below.

Vertical jump detection algorithm

- **NFR-01**: The video of the vertical jump to be processed must last a maximum of 2 seconds
- **NFR-02**: The vertical jump detection algorithm should detect the jump more than 95% of the attempts.
- **NFR-03**: The vertical jump detection algorithm must have an average error in the detection of the takeoff frame of less than 1 frame.
- **NFR-04**: The vertical jump detection algorithm must have an average error in the detection of the landing frame of less than 2 frames.
- **NFR-05**: The vertical jump detection algorithm must have an average takeoff and landing frame detection mean error of less than 1 frame.
- **NFR-06**: The vertical jump detection algorithm must have an average error in the vertical jump duration in frames of less than 2 frames.
- **NFR-07**: The vertical jump detection algorithm must have an average error in the vertical jump duration in frames of less than 25ms.
- **NFR-08**: The vertical jump detection algorithm must have a mean error in vertical jump height less than 3cm.
- **NFR-09**: The vertical jump detection algorithm should take less than 5 seconds to detect a vertical jump.
- **NFR-10**: The vertical jump detection algorithm must be implemented using OpenCV as a computer vision tool.
- **NFR-11**: The vertical jump detection algorithm must be implemented in Python.
- **NFR-12**: The vertical jump detection algorithm should return the results in JSON format.
- **NFR-13**: The data provided by the vertical jump detection algorithm after vertical jump detection must be:
  - Frames per second
  - Takeoff frame
  - Landing frame
  - Time in air in frames
  - Time in air in seconds
  - Height reached in meters
  - Peak power in W
  - Pear force in N
  - Initial velocity in meters/second

Vertical jump detector service

- **NFR-14**: The vertical jump detector service must be implemented using a REST API architecture.
- **NFR-15**: The vertical jump detector service must implement its REST API architecture using Flask and Python.
- **NFR-16**: The vertical jump detector service must be replicated 5 times.
- **NFR-17**: The replicas of the vertical jump detector service must be handled by an Nginx-based load balancer.
- **NFR-18**: The vertical jump detector service must implement JWT token authentication.
- **NFR-19**: The vertical jump detector service must offer the ability to upload the video to an AWS Bucket S3.
- **NFR-20**: The vertical jump detector service must be connected to a MongoDB base using the PyMongo library.
- **NFR-21**: The replicas of the vertical jump detector service and its load balancer must be deployed on an Amazon EC2 instance.

Backend service

- **NFR-22**: The backend service must be implemented using a REST API architecture.
- **NFR-23**: The backend service must implement its REST API architecture using Flask and Python.
- **NFR-24**: The backend service must be replicated 2 times.
- **NFR-25**: The replicas of backend service must be handled by an Nginx-based load balancer.
- **NFR-26**: The backend service must implement JWT token authentication.
- **NFR-27**: The backend service must be connected to a MongoDB base using the PyMongo library.
- **NFR-28**: The replicas of the backend service and its load balancer must be deployed on an Amazon EC2 instance.

Database

- **NFR-29**: The database must be implemented using MongoDB.
- **NFR-30**: The database controller must be replicated 2 times.
- **NFR-31**: The database controllers must be handled by an Nginx-based load balancer.

Android application

- **NFR-32**: The application must use the REST API created in the vertical jump detection service and the backend service.
- **NFR-33**: The data sent by the application to the REST APIs of the vertical jump detection service and the backend service will be sent in JSON format
- **NFR-34**: The lowest Android version that the application should be compatible with will be Android Lollipop (5.0) SDK version 21.
- **NFR-35**: The aesthetics of the application will follow the Material Android Design Guidelines.
- **NFR-36**: The application must use the Koin library to manage dependency injection.
- **NFR-37**: The application must use the Retrofit2 library to manage REST API requests and responses.

3.2.3. Business rules

Business rules are a specification of non-functional requirements that encompass behaviors and rules that the system must follow to meet quality requirements. In this system the following business rules have been extracted.

- **BR-01**: A vertical jump video is related to only one user.
- **BR-02**: A vertical jump is related to only one user.
- **BR-03**: A vertical jump is related to only one vertical jump video or none.
- **BR-04**: If the duration of a vertical jump video is longer than two seconds, it must be trimmed before making the request for its detection.
- **BR-05**: A vertical jump must be validated by the user after its detection to be shown to the user.
- **BR-06**: A vertical jump is related to only one vertical jump detector version.
- **BR-07**: If a video is successfully processed, must exist a vertical jump with that Video ID.

3.2.4. Information requirements

These requirements are a variant of the non-functional requirements to specify the information to be used in the system. The information requirements extracted for this system are as follows.

- **IR-01**: For each user will be stored:
  - User ID
  - Name
  - Surname
  - Email

  - Age
  - Height in cm
  - Weight in kg
  - Hashed password
  - Salt

- **IR-02**: For each vertical jump will be stored:
  - Vertical jump ID
  - User ID
  - Video ID
  - Detector version ID
  - Frames per second
  - Takeoff frame
  - Landing frame
  - Time in air in frames
  - Time in air in seconds
  - Height reached in meters
  - Peak power in W
  - Pear force in N
  - Initial velocity in meters/second
  - Valid jump
- **IR-03**: For each vertical jump video will be stored:
  - Video ID
  - User ID
  - Vertical jump video source
  - Successfully processed
- **IR-04**: For each vertical jump detector version will be stored:
  - Detector ID
  - Name
  - Description

3.3 Use cases

To complete the specification of the functional requirements made in subsection 3.2.1, we proceed to carry out a diagram of cases of the system. This diagram focuses on explaining the interaction of the entities that intervene in the system, which are normally:

- **Actors**: They represent the elements that interact with the system, from the users themselves to external systems. These actors represent a role, not specific entities.
- **Use cases**: They are the functionalities of the system, the functional requirements.

- **Relationships**: These represent relationships between actors and use cases or between use cases among themselves.
- **The software system**: Represents the part from which the functional requirements have been extracted.

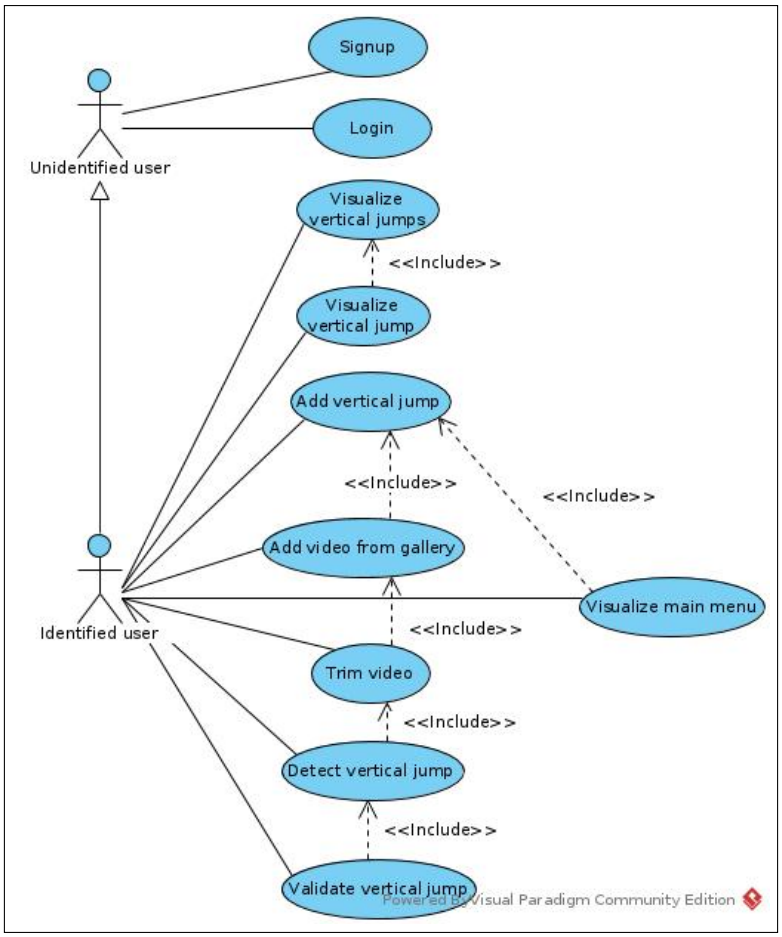


Figure 3.1: Vertijump use case diagram

In the diagram in the figure 3.1, it can be seen that there are two roles in the actors in the diagram, the identified user and the unidentified user. This division of roles shows the possible actions to be taken before logging in and once the user session has started. The use cases of this figure have been extracted from the requirements obtained previously, organized in a way that shows the relationships between all of them.

As you can see, there is a list of use cases that include the previous one. This is because they must be done in an order to be able to carry out the use case that encompasses all of them, add a vertical jump and obtain their data.

In addition, the flow to follow is shown to be able to see the details of a vertical jump since the use case that this action represents includes the use case to see vertical jumps, so it is necessary to enter the list of vertical jumps first to be able to visualize a specific vertical jump.

3.4 Conceptual modeling

The components that represent the system to be created have been designed based on all the aspects extracted from the analysis of the system. All these components of the conceptual modeling phase are of vital importance for the design and the rest of the stages.

The objective of this conceptual modeling is to show the main components of the system and their relationships. This is done also starting from the information requirements and the business rules of sections 3.2.4 and 3.2.3 respectively, illustrating it in the diagram in figure 3.2.

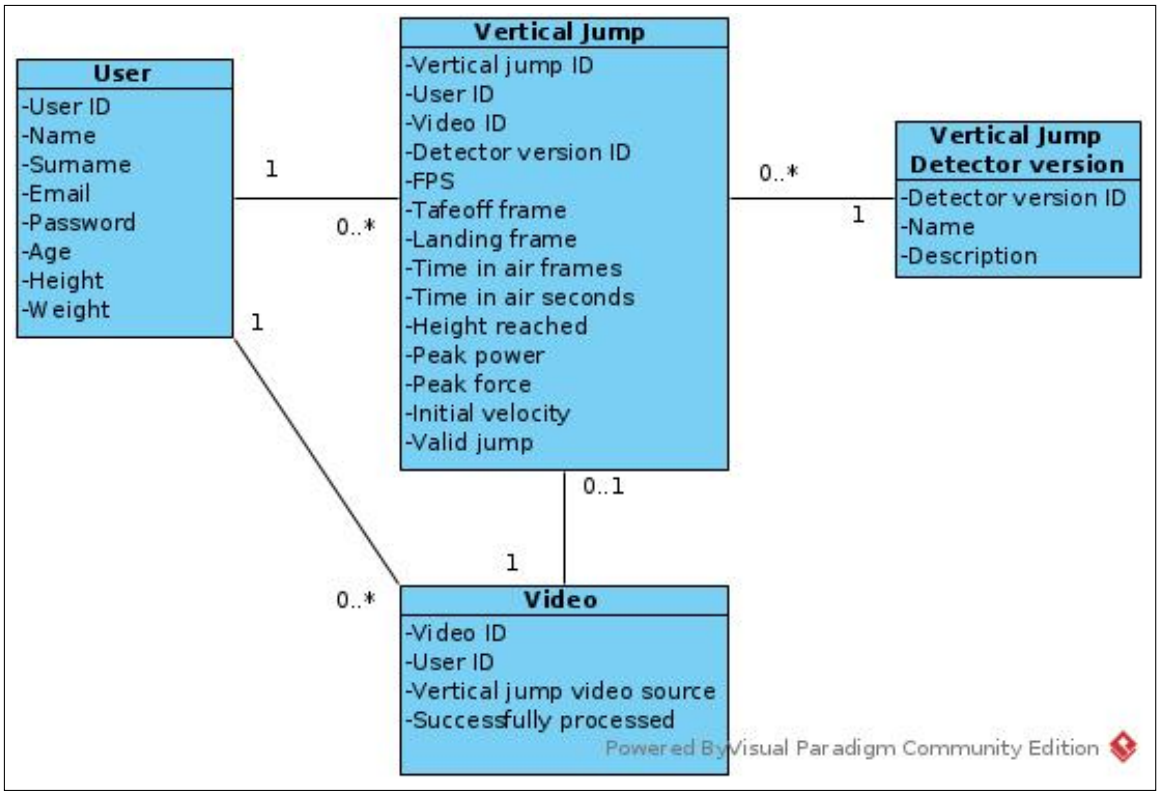


Figure 3.2: Vertijump conceptual model

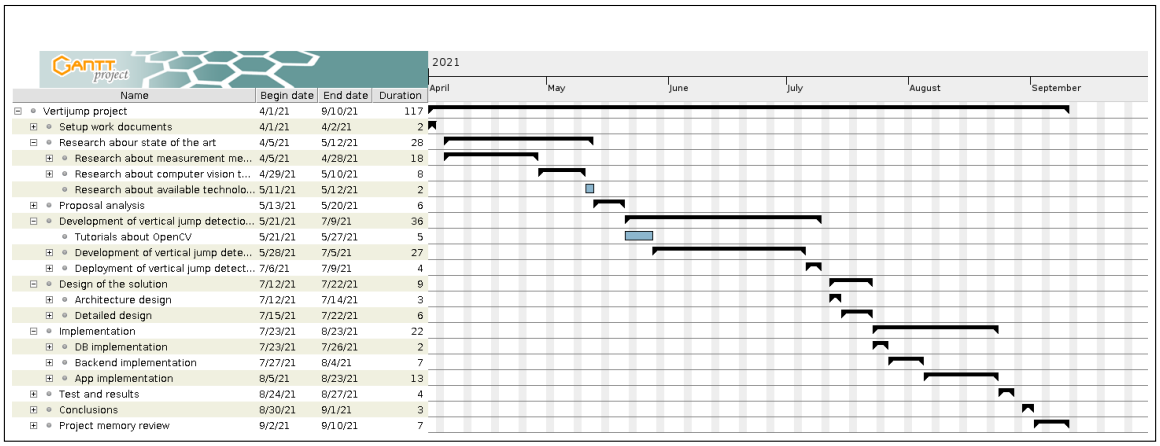
As can be seen in the diagram, the main component of the system is the vertical jump. These have relationships with the rest of the components of the system of the conceptual modeling stage. As the second most important component, the user stands out, since he can have relationships both with vertical jumps and with videos. It can be seen that vertical jumps always have an associated detector version. This allows that, if the jump is not validated, the versions of the detector that are presenting these failures can be obtained.

CHAPTER 4
Proposed solution

The basic ideas of the proposed solution have been detailed in the previous chapter, and after this analysis, this chapter first deals with the management aspects carried out for the detailed design of this project. This chapter is also identified with the design phase of the cascade methodology, so aspects such as the architecture or the design of the implementation of each of the parts of the system will be detailed in depth.

4.1 Project plan

The planning of the work of this project has been established according to the chosen methodology, together with some changes that were detailed in section 1.3, due to the requirement of a research and testing stage prior to the development of the rest of the application systems.



Name	Begin date	End date	Duration
Vertijump project	4/1/21	9/10/21	117
Setup work documents	4/1/21	4/2/21	2
Research abour state of the art	4/5/21	5/12/21	28
Research about measurement me...	4/5/21	4/28/21	18
Research about computer vision t...	4/29/21	5/10/21	8
Research about available technolo...	5/11/21	5/12/21	2
Proposal analysis	5/13/21	5/20/21	6
Development of vertical jump detectio...	5/21/21	7/9/21	36
Tutorials about OpenCV	5/21/21	5/27/21	5
Development of vertical jump dete...	5/28/21	7/5/21	27
Deployment of vertical jump detect...	7/6/21	7/9/21	4
Design of the solution	7/12/21	7/22/21	9
Architecture design	7/12/21	7/14/21	3
Detailed design	7/15/21	7/22/21	6
Implementation	7/23/21	8/23/21	22
DB implementation	7/23/21	7/26/21	2
Backend implementation	7/27/21	8/4/21	7
App implementation	8/5/21	8/23/21	13
Test and results	8/24/21	8/27/21	4
Conclusions	8/30/21	9/1/21	3
Project memory review	9/2/21	9/10/21	7

Figure 4.1: Vertijump project Gantt diagram

As shown in figure 4.1, the project begins on Monday, April 5, 2021, the day on which this idea is chosen and the work plan to be carried out begins. The project begins with the adaptation of the templates for the final master's thesis and the creation of the memory structure, thus establishing a base from which to document the entire process.

After this small set up, the first phase of the project begins, the analysis of the state of the art. This research phase provides a basis for the rest of the project, so it is carried out in a relaxed way using 28 days for its realization, analysis and documentation. It should be noted that it is divided into two main parts as can be seen in the figure 4.2, on the one hand the research on vertical jump measurement methodologies, and on the other hand the research on computer vision tools, which took 18 and 8 days, respectively.

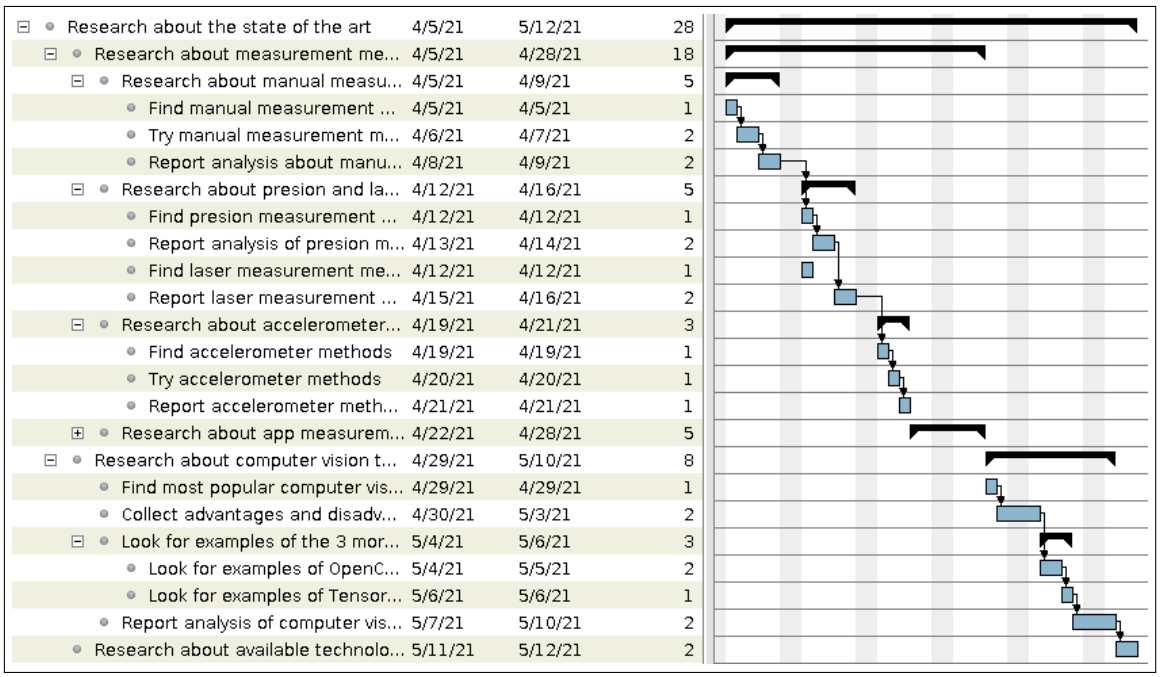


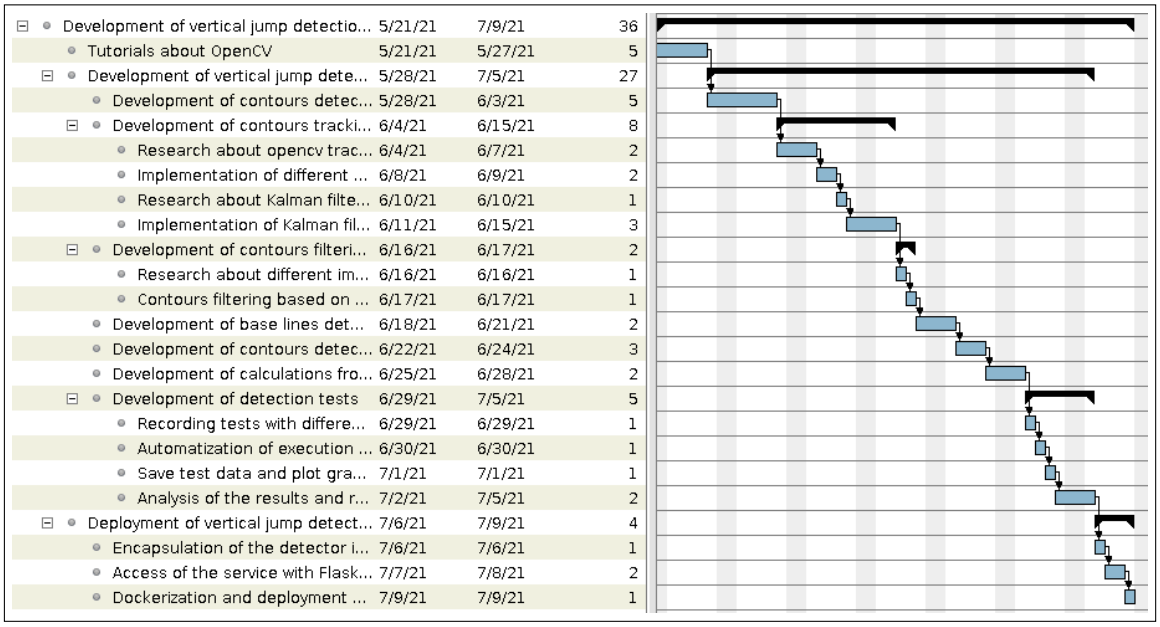
Figure 4.2: Vertijump State of the art Gantt diagram

Once this phase is finished, the analysis of the idea begins on May 13. This phase is responsible for the elicitation of requirements and the formation of an idea of the elements of the system and lasts for 6 days.

Then begins the phase that does not fit within the cascade development, the development of the vertical jump detection system through video. This stage is a process of investigation, trial and error to find a valid method to detect vertical jumps. It is divided into several phases as can be seen in the figure 4.3, in which different forms of detection are studied, ending all of these by finding a detection method that meets the quality requirements of the system. This phase is the longest since 36 days have been needed. It is the phase with the greatest error in the estimation due to the fact that a problem was faced with zero knowledge of the technology and no previous studies carried out.

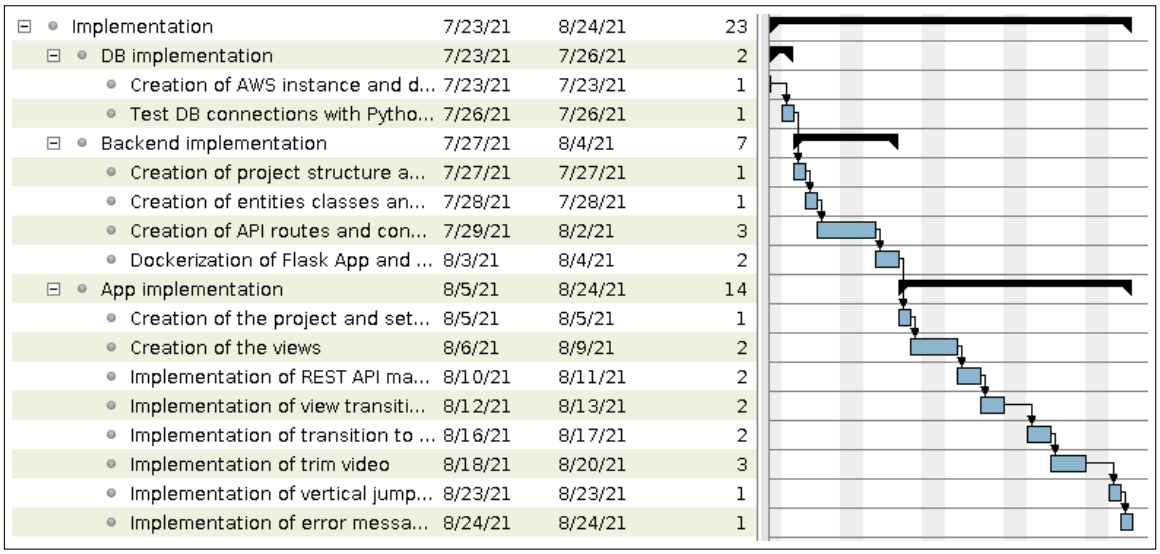
The next phase is the design of the application, carried out during almost 2 weeks, detailing the architecture and design in detail of both the backend and the application.

Once all these steps are completed, the implementation phase begins. It is one of the longest lasting 23 days, due to its complexity despite not having an abundance of functionalities as can be seen in the figure 4.4. This is due for example to using Kotlin without any prior knowledge.



Development of vertical jump detectio...	5/21/21	7/9/21	36
Tutorials about OpenCV	5/21/21	5/27/21	5
Development of vertical jump dete...	5/28/21	7/5/21	27
Development of contours detec...	5/28/21	6/3/21	5
Development of contours tracki...	6/4/21	6/15/21	8
Research about opencv trac...	6/4/21	6/7/21	2
Implementation of different ...	6/8/21	6/9/21	2
Research about Kalman filte...	6/10/21	6/10/21	1
Implementation of Kalman fil...	6/11/21	6/15/21	3
Development of contours filteri...	6/16/21	6/17/21	2
Research about different im...	6/16/21	6/16/21	1
Contours filtering based on ...	6/17/21	6/17/21	1
Development of base lines det...	6/18/21	6/21/21	2
Development of contours detec...	6/22/21	6/24/21	3
Development of calculations fro...	6/25/21	6/28/21	2
Development of detection tests	6/29/21	7/5/21	5
Recording tests with differe...	6/29/21	6/29/21	1
Automatization of execution ...	6/30/21	6/30/21	1
Save test data and plot gra...	7/1/21	7/1/21	1
Analysis of the results and r...	7/2/21	7/5/21	2
Deployment of vertical jump detect...	7/6/21	7/9/21	4
Encapsulation of the detector i...	7/6/21	7/6/21	1
Access of the service with Flask...	7/7/21	7/8/21	2
Dockerization and deployment ...	7/9/21	7/9/21	1

Figure 4.3: Vertijump Vertical jump detection Gantt diagram



Implementation	7/23/21	8/24/21	23
DB implementation	7/23/21	7/26/21	2
Creation of AWS instance and d...	7/23/21	7/23/21	1
Test DB connections with Pytho...	7/26/21	7/26/21	1
Backend implementation	7/27/21	8/4/21	7
Creation of project structure a...	7/27/21	7/27/21	1
Creation of entities classes an...	7/28/21	7/28/21	1
Creation of API routes and con...	7/29/21	8/2/21	3
Dockerization of Flask App and ...	8/3/21	8/4/21	2
App implementation	8/5/21	8/24/21	14
Creation of the project and set...	8/5/21	8/5/21	1
Creation of the views	8/6/21	8/9/21	2
Implementation of REST API ma...	8/10/21	8/11/21	2
Implementation of view transiti...	8/12/21	8/13/21	2
Implementation of transition to ...	8/16/21	8/17/21	2
Implementation of trim video	8/18/21	8/20/21	3
Implementation of vertical jump...	8/23/21	8/23/21	1
Implementation of error messa...	8/24/21	8/24/21	1

Figure 4.4: Vertijump Implementation Gantt diagram

Finally, shorter phases are carried out for the tests, analysis of conclusions and the review of the memory by the tutor and the developer of this project, ending on Monday, September 13.

4.2 Budget

4.2.1. Infrastructure and tools budget

In carrying out this project, only one person was in charge of all the tasks, so the expenditure on external personnel has been zero. The context in which it is developed is a master's project carried out by an unemployed student, which further reduces the budget. This is one of the reasons that have led to the use of mostly open source tools at no cost, avoiding expenses in technologies.

Apart from this, student plans have been used from some applications such as GitHub [1] or GitKraken [2], which offer a free PRO plan to student staff for product development without a commercial purpose.

Regarding the deployment of the system, it has been done with a free ASW account [3], which during the first year offers low-power resources at zero cost to learn how to use its systems. All systems deployed in this work have been deployed using devices of this free layer, so the cost of deployment has also been zero.

To avoid more expenses, any domain has been bought. Those provided automatically by ASW have been used, although they are less attractive.

Finally, the APK of the application is available through a GitHub repository, so the cost of its deployment is also zero due to not using a Google Play Developer account.

4.2.2. Programmer budget

After this analysis, the only cost that the project could have would be the hours used for its development. If we obtain the average salary of a junior programmer in Spain [4], we obtain a figure of € 18,953 gross per year for a 40-hour week, which gives a monthly salary of € 1,579 per month in 12 payments.

If the price/hour that is being paid to the junior programmer is extracted, we obtain a gross salary of € 9.86 per hour. The project has a total duration of 118 days and an average of 3.5 hours worked per day have been carried out.

This gives a total of 117 days * 3.5 h/day * € 9.86/h = € 4,072.18 gross price of the programmer salary.

4.3 System architecture

First, it is necessary to detail the general architecture that the system as a whole will use. An architecture in which tasks are divided into two groups, the resource providers, or servers, and the resource requesters, or clients.

As the world of software development progresses, these two parts are increasingly divided, giving rise to two main programming strands:

- **Backend**: This refers to the programming part of the server where all the business logic and access to persistence are located.
- **Frontend**: This part refers to the programming of the user interface that receives and displays the information to the user.

This division of roles is normally implemented, and as it is in this case, through a REST API architecture, in which the client (the application) requests data through HTTP requests to the server API (backend and vertical jump detector), the server processes the request, accesses the data persistence if necessary and returns the result through HTTP

[1] GitHub Student Developer Pack: https://education.github.com/pack
[2] GitKraken Student Developer Resources: https://www.gitkraken.com/student-resources
[3] AWS Free account capabilities: https://aws.amazon.com/free/?nc1=h_ls
[4] Spanish junior programmer average salary: https://es.indeed.com/career/programador-junior/salaries

protocol as well as the request as can be seen in the figure 4.5. This architecture is the most used both on webs and mobile devices due to it's simplicity and scalability [5].

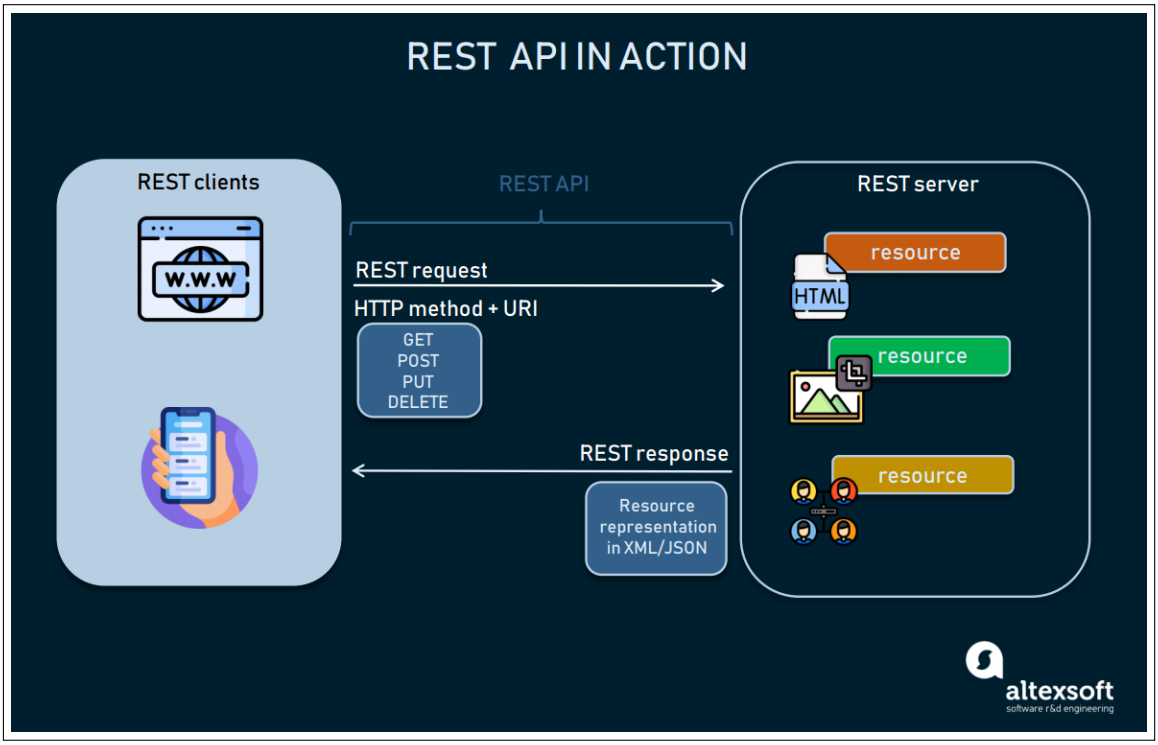


Figure 4.5: REST API architecture

4.3.1. Backend architecture

In this part of the project, because it is implemented using Python and Flask, a structure based on Blueprints is used but with a slight difference, adding a layer for access to the database and eliminating the part related to web pages, its design and its static components.

This structure is the most recommended for developing large Flask projects [6], dividing the code into modules which are in turn divided into layers. It is important to note that when using Python, an important point for its architecture is to use a virtual environment, which, as can be seen in figure 4.6, is also a part of the architecture. The layers to highlight of this architecture are:

- **Main layer**: This layer is located at the root of the project or service. It is responsible for hosting the rest of the layers and modules, along with hosting the starting point of the application *app.py*. The *setup.py* file, the configuration file of our Python application, is also located in this layer. If you want to add a library dependencies file, this is the layer where it should be, specifically through the file *requirements.txt*.

[5] What is and why use a REST API: https://www.pluralsight.com/blog/tutorials/representational-state-transfer-tips

[6] How to structure large Flask applications: https://www.digitalocean.com/community/tutorials/how-to-structure-large-flask-applications

```
~/LargeApp
    |-- app.py
    |-- config.py
    |__ /env              # Virtual Environment
    |__ /app              # Our Application Module
         |-- __init__.py
         |-- /module_one
             |-- __init__.py
             |-- controllers.py
             |-- models.py
         |__ /templates
             |__ /module_one
                 |-- hello.html
         |__ /static
         |__ ..
         |__ .
    |__ ..
    |__ .
```

Figure 4.6: Flask Blueprints architecture

- **Modules**: Following this architecture, modularization should be encouraged in order to have an orderly and scalable development. These modules in turn are responsible for hosting common elements in all of them, such as *models.py* or *controller.py*, which are detailed below. An example of this is seen in the module *module_one*. It is at this level also where the python package initialization file, *__init__.py* [7], should be added.

- **Module files**: This point is as extensive as the user wants, since files can be created that go in all modules to manage specific aspects such as forms, through a *forms.py* file. These files receive the same name in all modules and they perform the same function in all modules. Some examples of common files for all modules are:

  - *controllers.py*: This file is responsible for encapsulating the logic of the module. It is in this file where the routing of the REST API calls that request resources from this module is carried out. These calls normally use information from the entities modeled in models.py, so this file is in charge of using the models.
  - *models.py*: This file is in charge of encapsulating the classes that represent the entities of the module. These classes typically represent domain entities that may or may not be present in the database. It is in this file where the connection to the database is made to interact with the system entities.
  - *forms.py*: This class encapsulates the necessary forms in this module for the calls of the controllers that need it. This file does not communicate with any other of the aforementioned, it only provides forms to the module.

  This breakdown can go as far as the user wants, since it is possible in turn to create specializations of a file such as the controllers file, creating different controllers and routing the requests to them from the main controllers file. This structure helps a

[7] What is init.py: https://careerkarma.com/blog/what-is-init-py/

very high modularization that provides the system with much greater clarity and ease of maintenance.

4.3.2. Application architecture

For this part of the system, because it is developed in Kotlin, the chosen architecture is Clean Architecture [8]. This architecture allows the code to be separated into modules or layers as can be seen in the figure 4.7. These layers have restrictions to communicate with other layers, thus maintaining a single communication flow in the application. The layers that make up this architecture are detailed below:

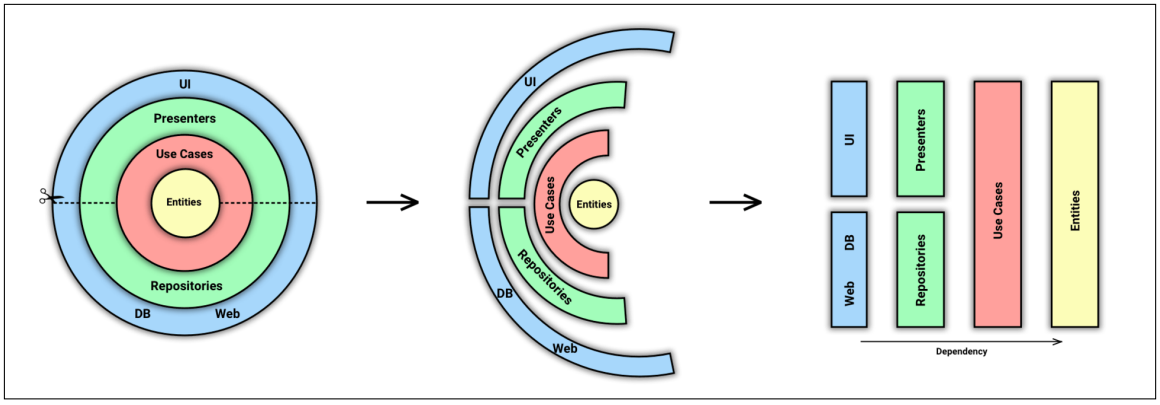


Figure 4.7: Kotlin Clean Architecture

- **Presentation**: This layer includes everything related to the views and graphic elements of the application and the actions that the user can perform on them. To implement this layer an MVVM pattern (model, view, view model) is used, it is very similar to MVC (model, view, controller). This model is made up of the following elements:
  - The model, as in MVC, represents the data layer and the business logic.
  - The view presents the information to the user and reacts to changes in the model, similar to an active MVC pattern.
  - The view model is an intermediary actor between the model and the view and contains all the presentation logic.

  The view in MVC depends on the model and the controller to get data or modify it. If there is a change in the view that implies a change in the model you need to access the controller and if there is new data, it depends on the model to display it. In the case of MVVM, the view only depends on the view model to get data or modify it, also no one has to tell it to update the data as the controller would in passive MVC, as it is observing changes in the model directly.

  The model in MVC can have too many responsibilities like getting the data from data sources, informing the controller about changes to that data, and preparing it for display in the view. In MVVM the model is completely decoupled from the view and only contains information, never actions to manipulate it.

  In MVC it is not very clear which part should be in charge of the presentation logic since the model should only send data to view and, on the other hand, bringing this

[8] Android Kotlin Clean Architecture: https://proandroiddev.com/android-clean-architecture-kotlin-flow-hilt-simplest-way-415d7e0f41b

logic to view would complicate some tests. At MVVM, this responsibility is very clear and falls into the hands of the sight model.

It should also be clarified that in the programming of this mobile application a single *Activity* is used as a coordinator of views and its function is only to configure and display them, that is, each view is a *Fragment* and they are managed by the *MainActivity*. Each Fragment has its own view model with its logic encapsulated in it. This layer only communicates with the Use Cases layer.

- **Use cases**: It includes all the logic necessary to execute the previously defined use cases. This layer receives the actions that the user can trigger. These can be active actions (the user clicks a button) or implicit actions (the application navigates to a screen). From this point on, all other necessary actions can be executed on a child thread, allowing avoid interface blocking. This layer only relates to the Data layer.
- **Domain**: It contains the definition of the business rules in the form of interfaces to be able to execute the actions of the use cases received from the Use Cases layer.
- **Data**: It corresponds to the implementation of the business rules previously defined in the Domain layer. Most of the logic is simply the request and data persistence. This layer only relates to the Repository layer.
- **Repository**: Defines the application's communication with data sources by implementing a repository pattern. For a given request, it is able to decide where to find the information, whether locally, such as the mobile database, or remotely, such as the REST API service. This layer only relates to the Local and Remote layers.
- **Local**: It implements all communication management with the database and CRUD operations to be able to save and retrieve data from the mobile device database.
- **Remote**: It implements all the management of the connection with the REST API to obtain and send the necessary data. Both the Local layer and the Remote layer implement the DAO pattern to access the data, the Local layer accesses the database and the Remote layer to the REST API, the DAO pattern proposes to fully encapsulate the logic to access the data, in this way, the DAO provides the necessary methods to insert, update, delete and consult the information.

4.4 Detailed design

This chapter details the design made to carry out the project, complying with all the specifications and requirements detailed in the previous chapters. The design made here establishes the points to follow for further development and implementation. During the following sections, the design aspects of each of the parties involved in the project are detailed in parts.

4.4.1. Database design

Like the rest of the parts of the project, it is necessary to make a final design of the database that the developed system will use. This system is the scheme to follow in the subsequent creation of the real database of the system, which is represented in the figure 4.8.

This database is non-relational, so the established associations do not correspond to relationships such as foreign keys in non-relational databases. Each table represents a

collection and the relationships between the tables are the keys that each one of the documents in the collections stores in order to be able to reference documents from other collections. Lastly, the primary keys represent the MongoDB *ObjectIds*, not a usual primary key.

The data types that appear in the collections are not the actual ones either, because the modeling tool does not support non-relational databases such as MongoDB. The *varchar* data type refers to a *String* type, while the *varbynary(1)* type has been used to represent a *Boolean* type.

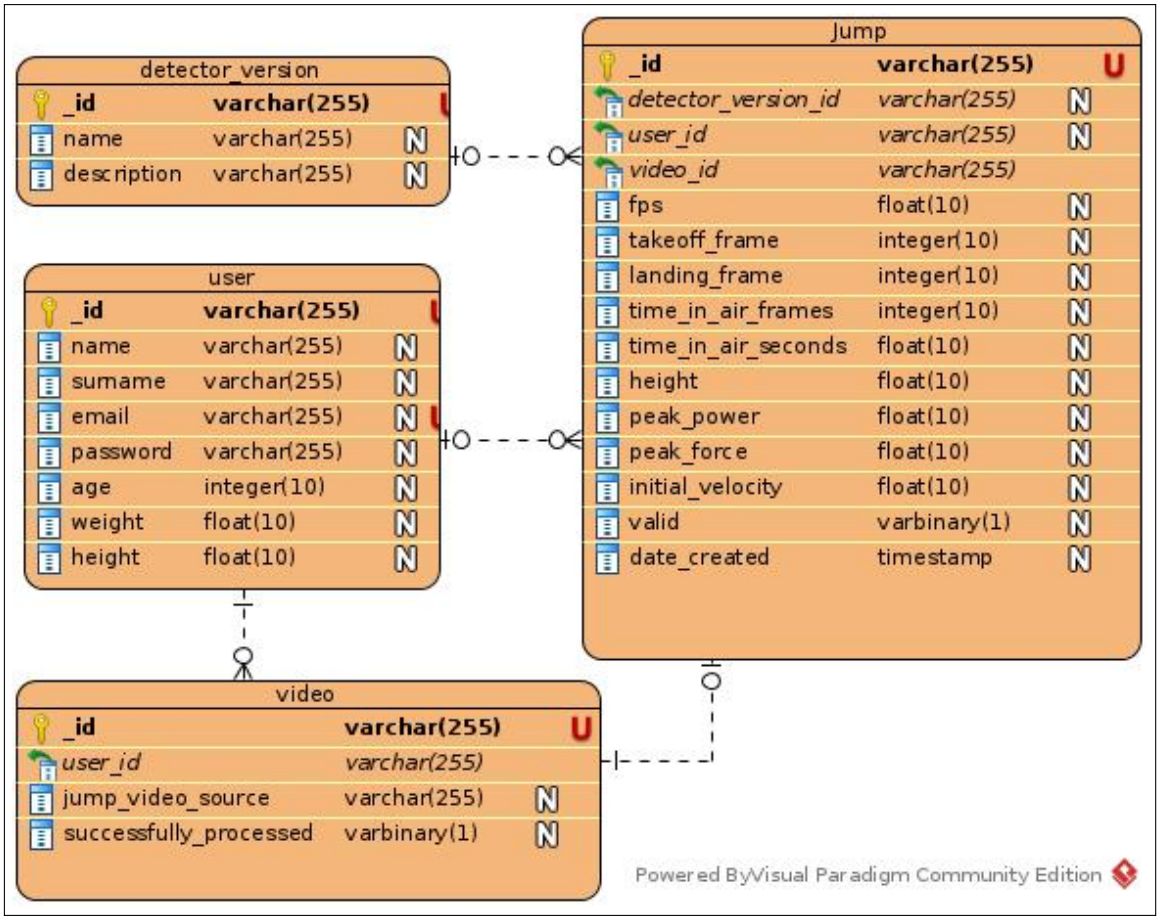


Figure 4.8: Database design diagram

As detailed in the analysis stage, the collection with the greatest importance is jump, since it has relationships with all the remaining collections. This makes sense since the system revolves around detection of jumps. Some of the jump fields could be calculated instead of stored, but this offers a speed advantage when it comes to testing and faster error or detection reports.

Regarding the attributes that require clarification, it is important to comment on two of them:

- Valid from jump: This attribute refers to an acceptance by the user of the jump results. The detection algorithm has not been tested in a full-scale test environment, so if the user determines that these results are invalid, the result is saved as invalid for further error analysis.
- Successfully processed from video: A video may return an error in its processing, from the non-detection of jump (due to its absence or due to a malfunction of the

detection) or because the video may be corrupted. This argument also serves for further investigation of detection or processing errors of the developed detector.

4.4.2. Vertical jump detector design

Although this part of the system could be treated together with the rest of the backend, it deserves to be treated separately from the rest since first, it is an autonomous component of the backend and second, its design entails the implementation of the vertical jump detection by computer vision, so its logic is not limited only to interacting with entities of a system and with the frontend.

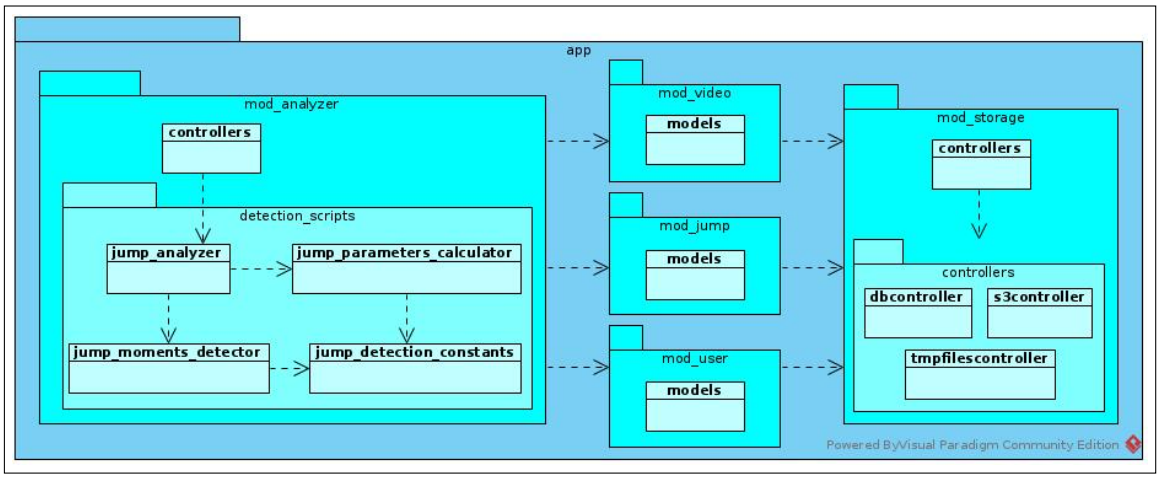


Figure 4.9: Vertical jump detector design diagram

As can be seen in figure 4.9, following the architecture detailed previously, the existence of a separate module is necessary to manage the detection of the vertical jump. This module has, like the rest of them, a router for processing requests, which redirects to the jump_analyzer class. This class, through the rest of its *detection_scripts* package, is in charge of first detecting the takeoff and jump moments of the video by means of *jump_moments_detector*, and second of calculating all the jump parameters using the logic implemented in *jump_parameters_calculator*. Both files use the constants present in *jump_detection_constants* to perform their tasks.

After this, the rest of the modules follow the structure set by the architecture but without having controllers, since this module is only designed to receive processing requests. The reason why the models are present is because after processing it interacts with some entities of the system to be able to save the analysis data in the database.

Finally, there are some comments about the *mod_storage* module. Video processing requires logic to be temporarily stored, logic implemented by *tmpfilescontroller*. In addition to this, the videos are stored in an Amazon S3 instance, so a controller has been created that encapsulates the logic of connections and interaction with the services of the S3 bucket. All of these controllers are managed by the main controller in the *mod_storage package*.

4.4.3. Backend design

The backend developed apart from the vertical jump detector does not present unusual or very abundant logic because there is not a very large amount of services to provide to the application. This makes the design follow the architecture almost to the letter, only

adding a *utils* module for the use of the modules that have common logic. Examples of this can be, for example, extracting the id of a user from its token, an implementation carried out by the *auth* module but which is made available to everyone from utils with another set of common functions.

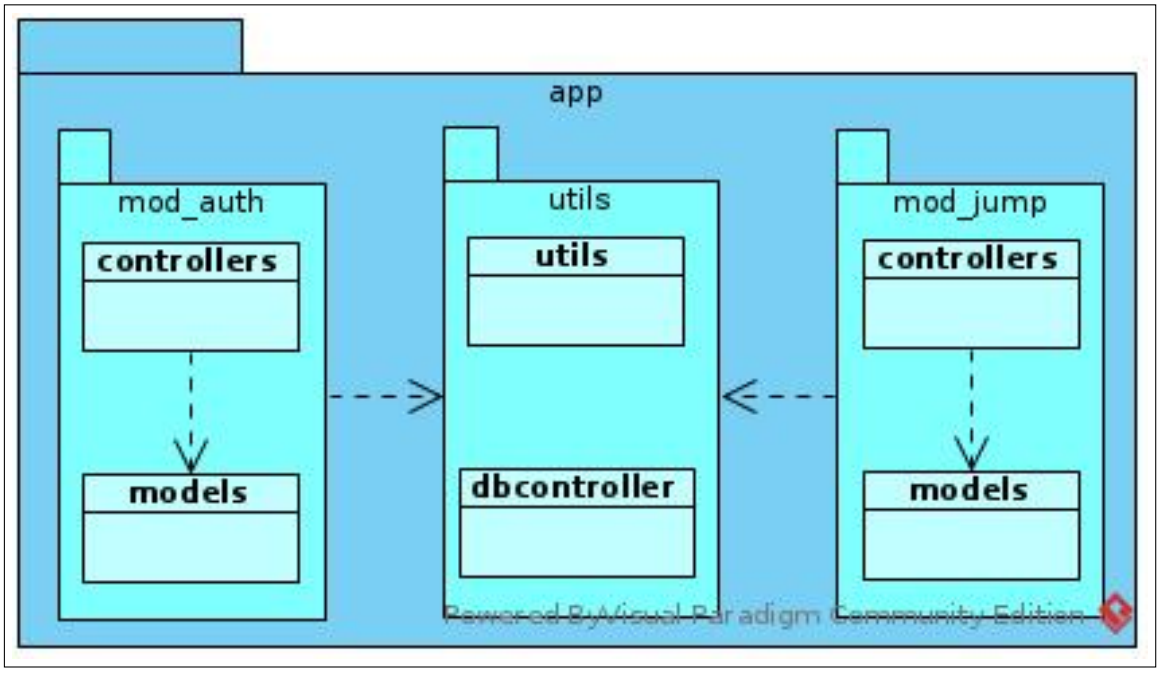


Figure 4.10: Backend design diagram

As can be seen in figure 4.10, the login and registration is implemented through the authentication module mod *_auth*. This module, following the architecture pattern, has a controller to route requests to authorization resources and a model to manage and detail the entities with which this module interacts.

Finally, there is the mod _jump module through which requests to jump-type resources are managed, and it manages the interaction with these entities through the models created in this module.

4.4.4. Application design

Contrary to the simplicity in the design of the backend, the design of Android applications is totally the opposite in terms of quantity of elements. Following the Clean Architecture, the same concept spans a diversity of layers in order to isolate the logic belonging to each one of them. For modularity it is a very strong point, but this results in a huge number of diagrams, so only two of the layers of the architecture are going to be shown.

The first of the detailed layers is the UI layer, one of the outermost layers of the architecture. As can be seen in figure 4.11, modules have been created for each of the elements of the graphical interface. Each of these elements has a Fragment [9], which represent a reusable part of the user interface and are executed in the *FragmentActivity*. After this, it can be seen that most also have a *ViewAction*, a *ViewModel* and a *ViewState*. These are the elements of the MVVM architecture, so the implementation of this architecture is demonstrated here. The mission of each of these elements is detailed in the subsection 4.3.2.

[9] What is a Fragment? https://developer.android.com/guide/fragments

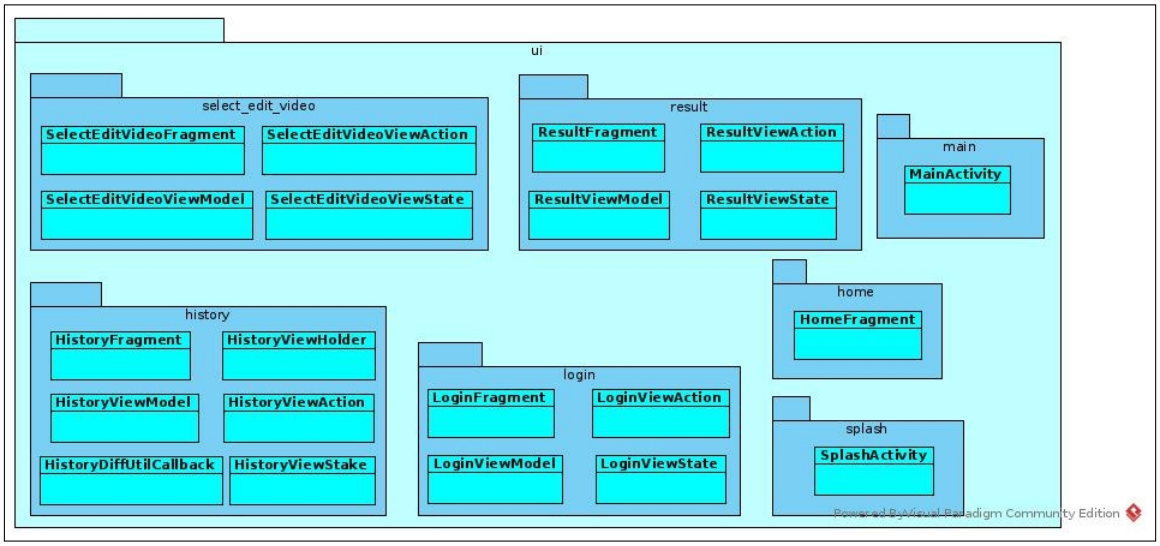


Figure 4.11: Application UI layer design diagram

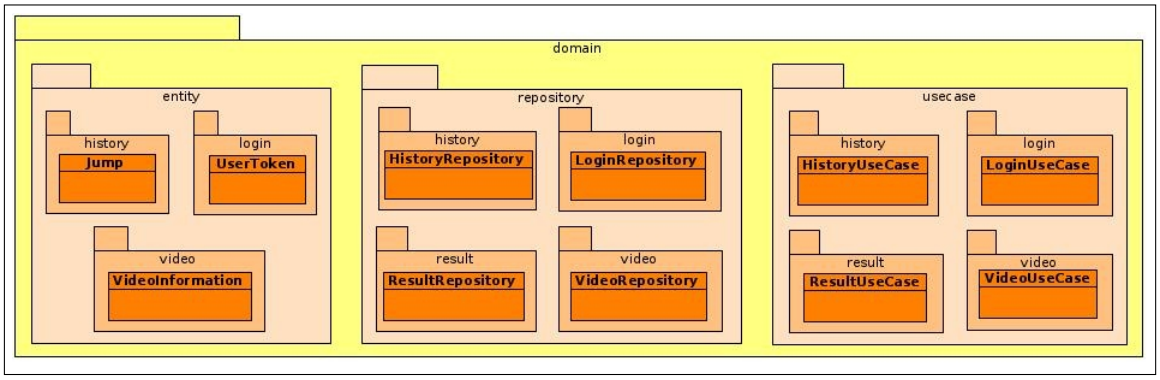


Figure 4.12: Applicacion domain layer design diagram

The second layer to be detailed is a simpler one, the domain layer, presented in figure 4.12. In this layer we can see how the elements of related layers are encapsulated, such as the repository layer or the use cases layer, layer with which it is connected through the business rules implemented in each of the elements of these modules. In this case we cannot find anything outside the architecture either, so the modularity is maximum and the names of the elements are repeated for each of the layers, isolating their operation.

4.4.5. User interface flowchart design

An important aspect to design is the flow that the user interface will follow with user interaction. This marks several business rules to be followed by the application, since it is detailing the actions allowed by the user in terms of navigation through the system. This navigation is detailed in the flowchart illustrated in figure 4.13.

This flow is detailed from the analysis stage, specifically the functional requirements detailed in subsection 3.2.1. It can be seen that there are several functional requirements that have a requirement as a dependency, so it can be interpreted as a previous step for the realization of the use case. Following these dependency flows leads to the flowchart detailed in this subsection.

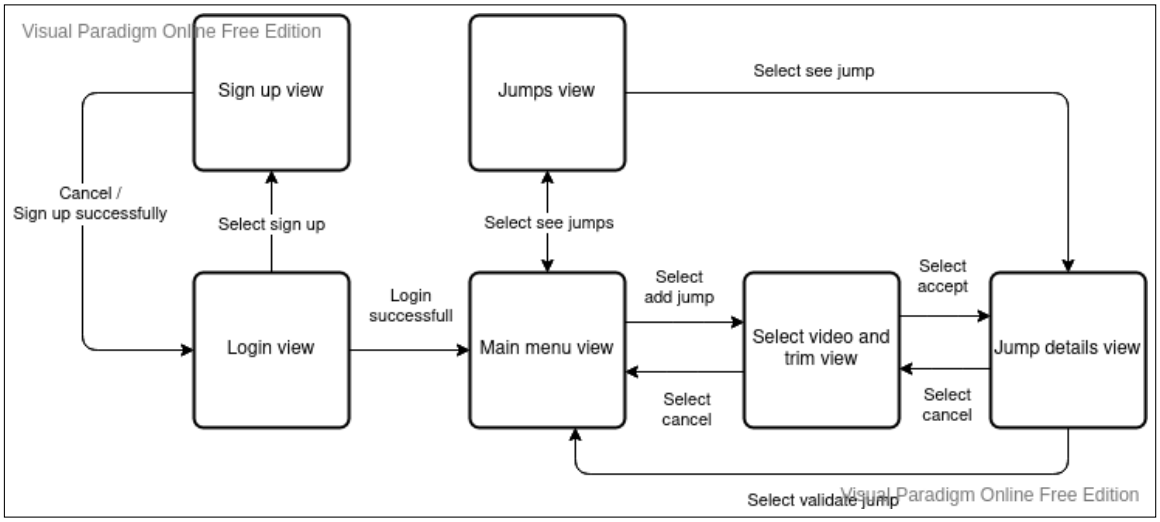


Figure 4.13: Application flowchart design

An important aspect to highlight is that, after signing up, it is necessary to log in. This behavior tends to vary a lot between other systems, but this time it is necessary to log in as any other user after registering. Another important aspect is that after validating a jump, you return to the main menu. This allows the user to reuse any of the system's capabilities automatically after jump processing.

4.5 Development of the proposed solution

In this section, the process of developing the solution proposed during this chapter and analyzed in the previous ones is detailed in parts, and in a way that does not lose focus on the details.

4.5.1. Development of the vertical jump detection system

In terms of complexity of development, this part of the project is the most complicated and with less clarity in terms of the duration of its development and the quality that could be expected from the result obtained. This is because more than a conventional development process, it is a research process about a topic that nothing was found on the networks or scientific articles.

This high complexity, together with the ignorance of the field of computer vision, made the granularity of the tasks very small in order to establish a knowledge base and review well each step that is taken, since a failure in this phase can lead to going down a wrong path that does not lead to any solution, as has happened at some time in development.

Project creation and version control

The first thing that is done is the creation of a project in GutHub to store the different versions that are carried out in order to find a solution to the detection. This particular project is called VertiJump-detector, as can be seen in the figure 4.14.

After creating the develop branch of the project, a Markdown document is added to record the specifications of each version developed. Apart from this, specific branches

are created for testing specific methods. Being a single user, code conflicts are very rare, but it is a way to leave a trace of the development and to be able to go back if a version does not work as expected.

Figure 4.14: Vertijump-detector repository

Contours detection algorithm

The first step to be able to detect an object in a video is to be able to detect moving objects in the video, which is translated into contour detection in a video. OpenCV is very famous for the typical tutorials for detecting cars or people in motion, so it is possible to perform an analysis of these algorithms. After an analysis of the algorithms, the steps carried out for the detection of contours are:

1. An OpenCV video capture is created with the path to the video to be processed or to the camera from which the images are extracted.
   cap = cv2.VideoCapture (source)
2. The first two frames of the video are extracted and a loop is started that lasts as long as there are frames to be read from the video grabber.
   ret, frame1 = cap.read()
   ret, frame2 = cap.read()
3. An absolute difference between the two frames extracted at the beginning of the process is calculated using an OpenCV function to subtract the pixels from two frames.
   cv2.absdiff (frame1, frame2)
4. The difference of the two frames is transformed to a grayscale.
   cv2.cvtColor (frame, cv2.COLOR_BGR2GRAY)
5. A Gaussian blur is applied to the grayscale image. This step differs greatly from one tutorial to another, since the choice of the kernel and the number of iterations of the dithering is up to the user, although it is a parameter that can make a total difference.
   cv2.GaussianBlur (frame, (7, 7), 5)
6. Thresholding is applied to the blurred image. This step makes pixels greater than a threshold set to their maximum value and those below a threshold to be set to their minimum value. Doing this on a grayscale image causes the image to go completely to absolute black and white. In this case, the user also decides from what value to apply the threshold, a value that together with the choice of blur can be key.
   cv2.threshold (blur, 20, 255, cv2.THRESH_BINARY)
7. A dilation is applied to the previous image. This transformation simply expands the edges highlighted with the previous processes more, making a contour with thinner parts more uniform.
   cv2.dilate (frame, None, iterations = 2)
8. Finally, after these transformations, a processed image is obtained to which the OpenCV function is applied to obtain the contours in its image.
   cv2.findContours (frame, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

After analyzing the steps that are applied to transform the frame, in figure 4.15 you can see each of the transformations applied to the image from its original version to the dilated version with the borders of the moving object marked.

Once the image contours have been obtained, they are iterated over and, after passing an area filter to verify that they have a minimum area, a square is drawn around each of the detected contours, thus marking all of them on the original image as can be seen in the figure 4.16. These squares are drawn using the OpenCV function *cv2.rectangle*, which needs the points of the upper right corner of the outline, their height and their width. These parameters are in turn extracted by another OpenCV function, specifically with the *cv2.boundingRect(contour)* function.

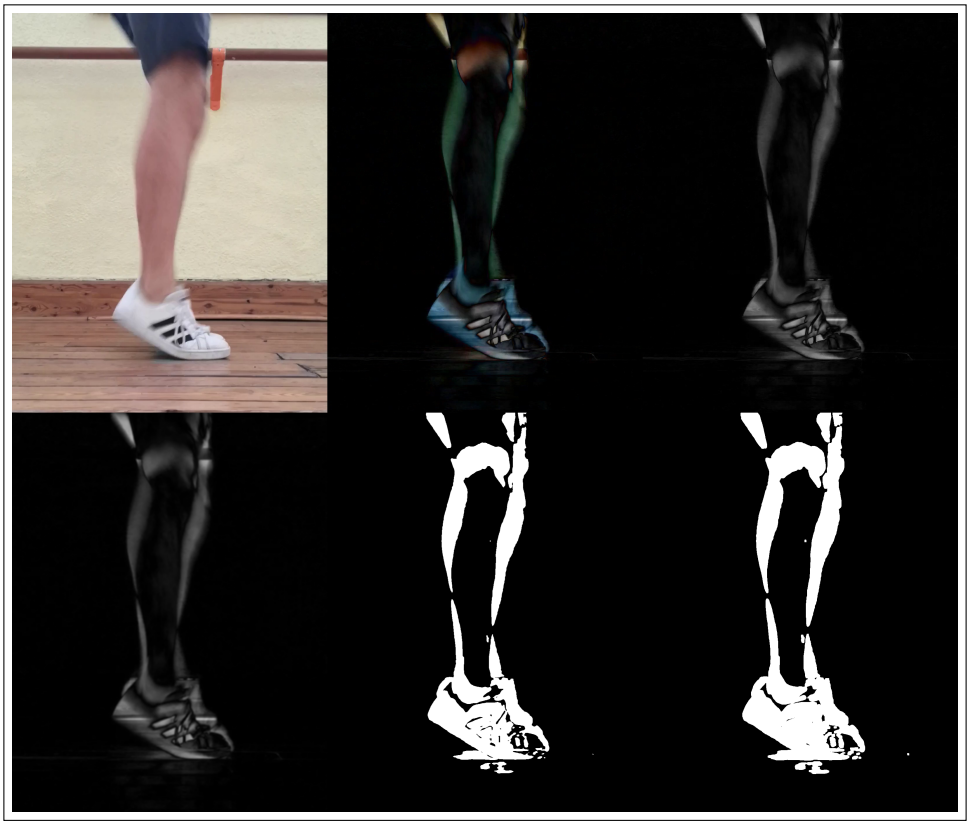


Figure 4.15: OpenCV image transformations

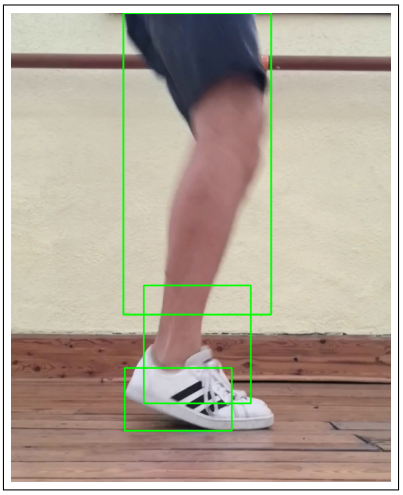


Figure 4.16: OpenCV contours detection

Tracking-based vertical jump detection

The first idea that comes to mind when trying to detect a jump is to track the feet of the subject to observe their movement and to be able to detect when the jump has been made quite accurately. This is a good idea based on good tracking performance, good contour processing, and accurate foot detection. The development of this solution presents problems from the first moment, in particular these are the most prominent:

- **Foot detection**: OpenCV has many libraries for the recognition of objects or faces, but these usually need a training model for the detection. There are models trained for faces, hands, gestures but not a model for feet with shoes in performing a vertical jump.
- **Tracking the object**: Even avoiding the problem of detecting the feet by manually selecting the area to be tracked, as it is a changing object, the tracker has a very low precision. Your feet move a lot during a vertical jump, even if you try to keep your foot as still as possible. To try to solve this problem, different types of trackers [10] can be used since OpenCV has a considerable number of options, but many of them have very slow processing or sacrifice precision to obtain a faster result.
- **Jerky movements of the tracker**: This is a reason derived from the second problem, but when trying to evaluate sudden movements of an object (moment of takeoff and landing), that the tracker moves abruptly [11] and not with a smooth tracking makes it very difficult, not being able to consider which movement is a jump and which is not.

After testing with different types of tracker implementations and, after analyzing their problems at the time of detection, this option is discarded in order to obtain an accurate result in the detection of vertical jump.

Contours baseline based vertical jump detection

After observing that the trackers are not a viable option, the option is proposed of saving the baselines of the detected contours and filtering them in some way to only obtain it when the contour may belong to a foot. This is another valid hypothesis, but for it to be proven, several requirements must be established for the video to be processed:

- **Distance to the camera**: In order to establish frames of which contour can be a foot and which cannot, it is necessary that the jumps be made at a similar distance in all the tests. This requirement opens the possibility of establishing a minimum and maximum area in which an average foot would enter, establishing a margin of error for those people who present a foot size greater or less than the average. By establishing a range from 1m to 1m 25cm, a large stride of an average person, a distance is established that allows this requirement to be applied.
- **Duration of the video**: This requirement is given because most people do not have someone to record them jumping, but rather they leave the phone supported and they perform the jump in front of the camera. This behavior causes the user to walk in front of the camera to get to the place of the jump, a behavior that can confuse

[10] Analysis of OpenCV trackers: https://www.pyimagesearch.com/2018/07/30/opencv-object-tracking/

[11] Object tracking problems with OpenCV: https://ehsangazar.com/object-tracking-with-opencv-fd18ccdd7369

the algorithm, since it is intended to detect baselines, contour lines detected as low as possible in the recording frame. Shortening the video to *2 seconds* eliminates this phase of hopping in front of the camera, along with the subsequent phase of going back to the recording device.

- **Background of the video**: Since a foot is no longer being sought, but rather a contour that falls within the norms of being a foot, it is necessary that no more movements are occurring in the background. This is fixed by placing the restriction of recording the video looking towards a wall, thus eliminating any movement from the bottom that could affect the detection of the vertical jump.

Once the conditions for video recording have been established, the baseline detection process the video as follows:

1. The detected contours are filtered according to a minimum area percentage and a maximum area percentage. If it is not within this range, that contour is not used for subsequent steps. Both percentage limits are set in a detector constants file so that they can be changed and further tested.

2. If the contour has a baseline less than the lowest saved plus an error range, the Y coordinate of the baseline of the rectangle surrounding that contour is set as a new baseline. The error range is also stored in a detector constants file to be able to carry out tests with different error margins.

3. If the contour baseline is not less than the current one plus the error range, it is checked if it is within this range. If said contour is within the range of the current baseline, the index of the current frame is added to the list of frames with contours detected in the detection range of the current baseline.

4. After going through all the frames and having saved all the possible baselines, the baselines are filtered as to whether their lists of frames with contours detected in their range are valid or not. To evaluate whether a list of frames is valid or not, minimum and maximum values of the duration of a vertical jump are also used, the minimum being a value well below the average and the maximum a little higher than the world record. This filtering is performed first, calculating the difference between each pair of consecutive detected frames and then filtering these differences with the previously established range. Here's an example:

    - Minimum jump frames duration: 12.
    - Maximum jump frames duration: 70.
    - List of frames with contours in the baseline range detected: [13, 15, 16, 45, 46, 47, 50, 52].
    - List of differences for each pair of frames: [2, 1, 29, 1, 3, 2].
    - Filtered list of differences in the valid range: [29].

5. Once the differences are filtered, it is enough to find the index of the last difference of the filtered list in the list of differences and with that index, the index of the frames between which there is that difference.

This process is performed for all detected baselines, so in the end there is only one set of baselines that have jump candidates. In this case, the last valid baseline detected is chosen, since being such a short duration of the video it is difficult for an event of that duration to occur after landing the jump.

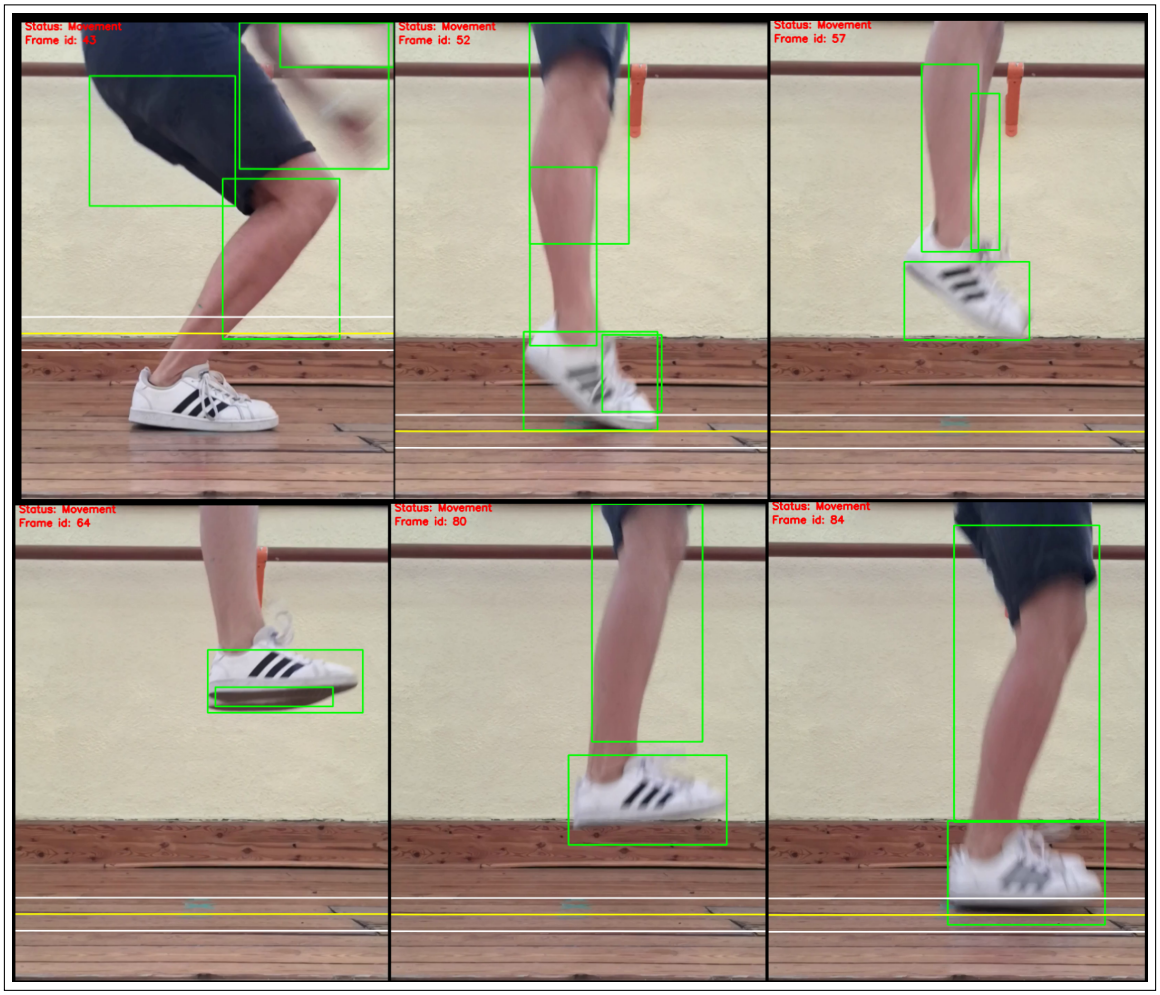


Figure 4.17: OpenCV baselines and contours detection

During the investigation and testing stage, all these lines are marked in the original image as can be seen in figure 4.17. Specifically, each line represents the following:

- Green lines: Contours detected.
- Yellow line: Current baseline.
- White lines: Upper and lower margin of the baseline contour detection range.

After having detected the vertical jump, the only thing that the detector has to return to the system when executing is a tuple of the take-off frame and the landing frame of the vertical jump.

Calculation of the attributes of a vertical jump

Once the take-off and landing frames and the jump video are available, the rest of the jump parameters are calculated. These parameters are all calculated based on the time in the air of the vertical jump. The parameters and their calculation method are detailed below:

- **Time in the air in frames**: Simply subtract the landing frame minus the take-off frame.

- **Time in the air in seconds**: For this attribute it is necessary to consult the video again to obtain its frames per second. There is an OpenCV function available for this, specifically through the *cap.get(cv2.CAP_PROP_FPS)* function. Once the frames per second are available, it is enough to divide the air time in frames between the frames per second.

- **Initial speed**: Knowing that this movement is a projectile motion, and having the time in the air in seconds, the initial speed of the jump can be calculated using the formula $t = 2 * Vo/g$, where Vo is the initial speed of the jump and g the force of gravity ($9.81 m/s^2$).

- **Height reached**: Taking the data of the initial speed and the force of gravity, the maximum height can be calculated using the formula $Hmax = Vo^2/(2 * g)$.

- **Peak power**: The maximum power is the one exerted at the moment of the jump, but to calculate this power, as there are no pressure plates, two different formulas are used and the average value between the two is calculated. These formulas are Harman, which has good results crossing it with formulas like Lewis's formula [12], and Johnson & Bahamonde [13]. The first uses only the user's weight and the second also uses height, so an average of both is returned.

- **Peak force**: By having the maximum power and speed in that instance, the maximum force is calculated as the $Pmax/Vo$, the speed of this moment being the initial speed.

Service development

After already having the vertical jump detection algorithm and a function that calculates all the parameters of the jump after the tuple returned by the detection algorithm, the detection algorithm is encapsulated so that it can be used by means of a REST API. This is done following the design made previously, creating the classes corresponding to the design and implementing those services.

A notable part of this service is the communication with an AWS S3 Bucket. The connection with this service has been implemented through the Python boto3 library [12], which through credentials and a URL allows you to access a library of functions to manage that storage space from Python. The connection established through this library is detailed in the code snippet 4.1.

```
S3_BUCKET_NAME = "vertijump-jumpvideos"

s3 = boto3.resource(
    service_name='s3',
    region_name='us-east-2',
    aws_access_key_id='AKIA3FW6Q2BZ6UT5EK',
    aws_secret_access_key='qsOGPjZr21VGXuyTvx5QAScpxvVbk8WsnzSnP'
)
```

Code Listing 4.1: S3 Bucket connection parameters

It is important to highlight that the uploading of the videos to this platform has been implemented as an asynchronous process through the multiprocessing library. This is due to the fact that the upload of the file is a blocking operation and it is only delaying the waiting time, because the analysis or any other part of the code needs the result,

[12] Introduction to Pyhon's Boto3 library: https://towardsdatascience.com/introduction-to-pythons-boto3-c5ac2a86bb63

only the instruction to delete the video from the temporary folder. The creation of the non-blocking process for the video upload is shown in the code snippet 4.2.

```
p = Process(target=asyncUploadVideoToS3, args=(
        file_source, file_bucket_source,))
p.start()
```

Code Listing 4.2: Python Process creation - S3 upload file

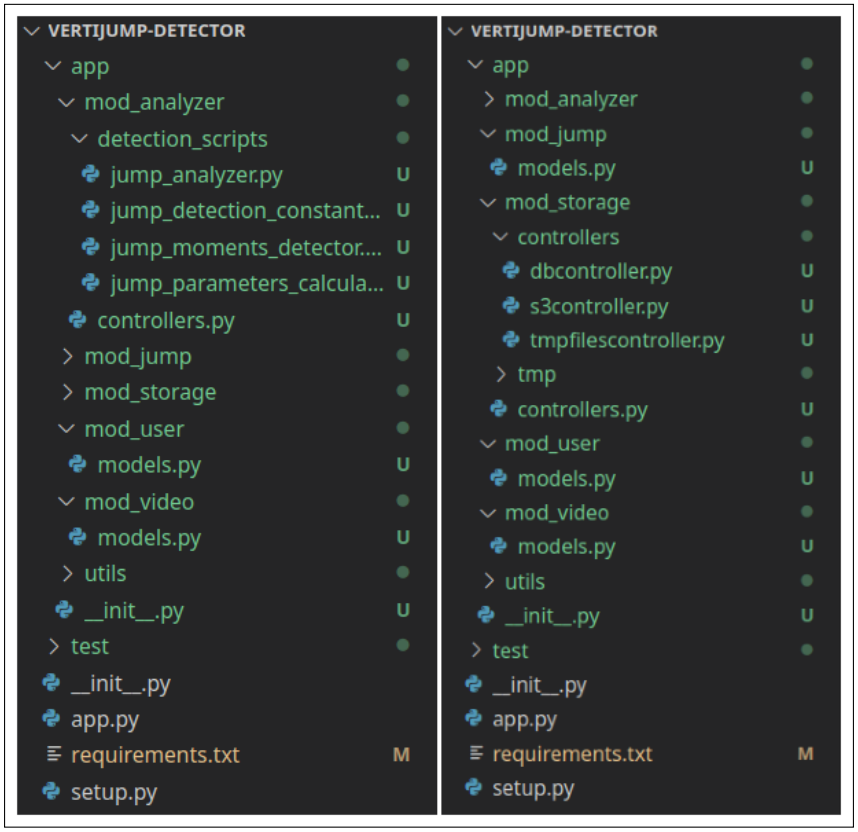


Figure 4.18: VertiJump detector code structure

After these explanations, in the figure 4.18 it is possible to see how the organization of the project remains following the detailed design and the structure marked by the architecture.

4.5.2. Development of the backend

This part of the project is much smaller and implements a very simple logic, authentication service and queries about system entities, without any type of processing or function outside of what is established in the basic architecture and its design in detail.

Project creation and version control

To develop this project, a new repository has been created, specifically the Vertijump-backend repository, as can be seen in the figure 4.19. Being a pure development part, the versions are managed through GitFlow, creating branches for the new features and merging to develop and later to master when tested.

Figure 4.19: Vertijump-backend repository

Development of the modules

The development has been done following the backend architecture tutorial itself. After the creation of an authorization module, a token authentication has been implemented, specifically using JWT. Thanks to this tool, a security layer is added to the backend, since without the token and its automated verification, which can be seen in the code snippet 4.3, it is not possible to access more resources than the login or registration. To add even more security, the password is kept encrypted, also creating the logic to compare a plain text password with an encrypted one on a specific user.

```
@app.route("/user/jumps", methods=["GET"])
@jwt_required()
def getUserJumps():
    return jump.obtainJumpsFromUser(request)
```

Code Listing 4.3: Python Flask JWT required - User jumps request

After the creation of the authentication module, the jump and user modules are created to implement their requests and their models, as can be seen in the code snippet 4.4.

```
class User(Document):
    name: StringField(required=True)
    surname: StringField(required=True)
    email = EmailField(unique=True, required=True)
    password = BinaryField(required=True)
    born_date = DateTimeField(default=datetime.utcnow)
    weight = FloatField()
    height = IntField()

    def __init__(self, name, surname, email, password, born_date, weight,
        height):
        self.name = name
        self.surname = surname
        self.email = email
        self.born_date = born_date
        self.weight = weight
        self.height = height

        self.password = hash_password(password)

    def get_id(self):
        return self._id

    @staticmethod
    def check_password(password_hash, password):
        return check_password_hash(password_hash, password)
```

Code Listing 4.4: Python User class

The rest of the system has continued to be completed according to the proposed design and architecture, so after completing its development, the code structure shown in figure 4.20 is obtained. A notable aspect in this development is the use of the

PyMongo library [13] to the connection to the database. It is the same library as in the previous case since it is the official MongoDB for Python.

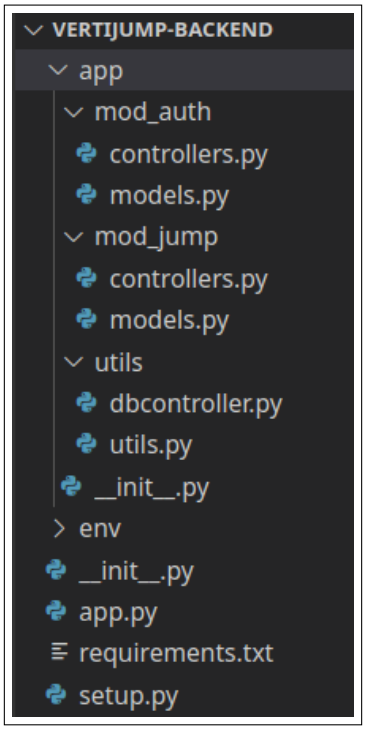


Figure 4.20: Vertijump backend code structure

4.5.3. Development of the application

This last part of the project is one of the largest in the system, due to its number of layers and resource consumption by the Android Studio IDE [14]. Development has been very slow due to late builds of the project.

Project creation and version control

The project has been created in Android Studio from an example of authentication in Android made with Kotlin. To control its development, a repository has also been created on GitHub, specifically called Vertijump-app.

Development of the modules

Like the backend, the development of the application has been mostly a follow-up of the architecture and the design in detail carried out, which is represented in the packages structure of the project, as can be seen in the figure 4.21. The main development aspect that should be highlighted is the use of the FFmpeg library for Kotlin [15], which has made it possible to implement the clipping to the video of the jump directly in the application for processing. After this, other challenges may be the implementation of token authentication and waiting for the jump results through a loading screen.

[13] How to use PyMongo: https://www.bmc.com/blogs/mongodb-pymongo/
[14] Android Studio system requirements: https://www.kindacode.com/article/android-studio-system-requirements/
[15] FFmpeg GitHub repository: https://github.com/fourtalk/ffmpeg-android-kotlin

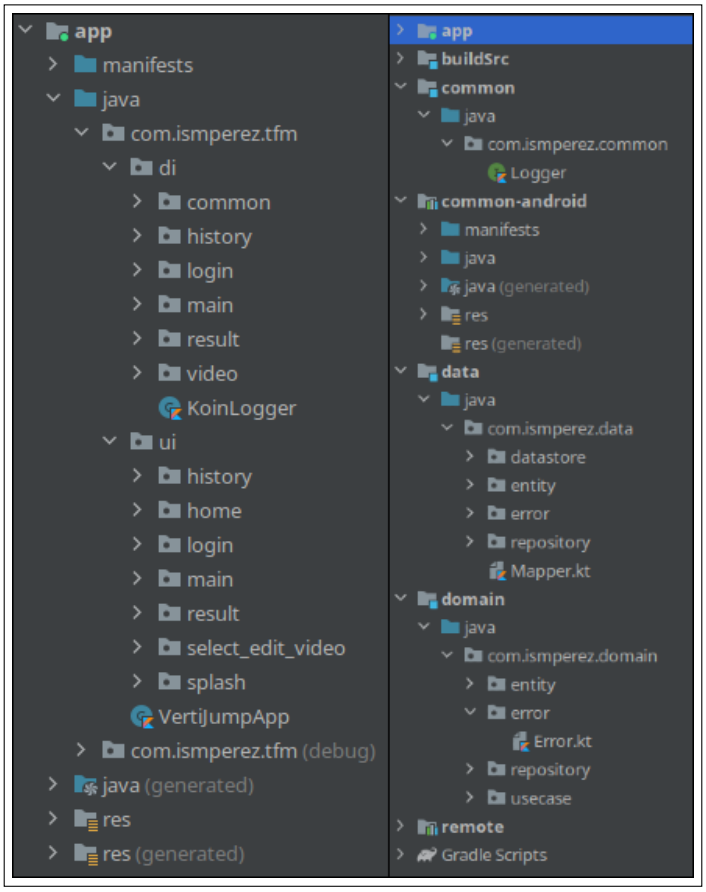


Figure 4.21: Vertijump app code structure

Development of the views

Views are the first aspect to build in the application. Its design has been carried out based on the need of the user cases and the Android Material Design Guidelines [16] have been followed, which can be seen in figure 4.22. These views have been implemented using Fragments, which are presented using the Coordinator Layout.

[16] Material Design Guidelines official webpage: https://material.io/design/guidelines-overview

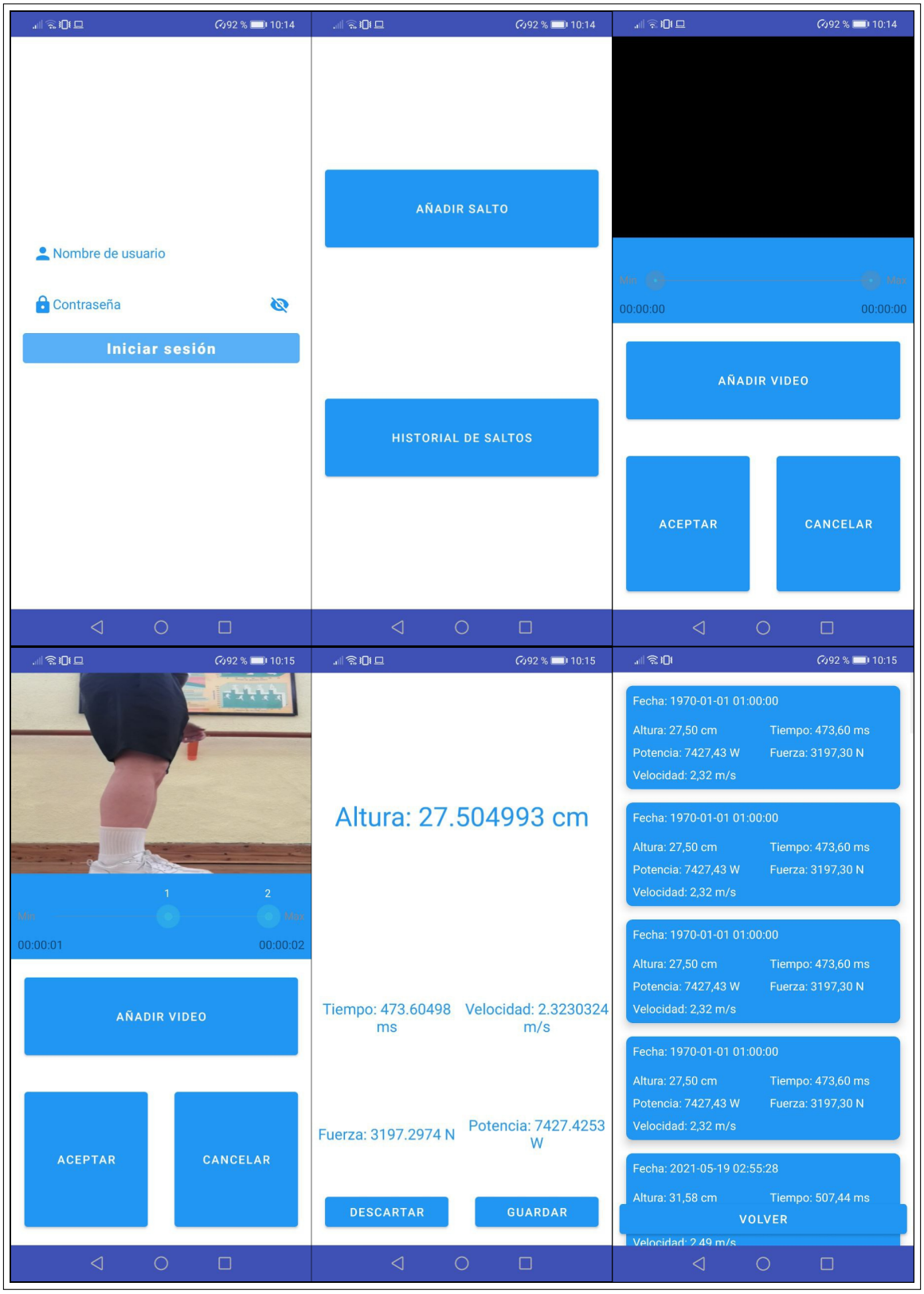


Figure 4.22: Vertijump app views

CHAPTER 5
Implantation

One of the purposes of this project is to test the idea as a minimum viable product of a business idea, so it is necessary to be able to use the solution in a real test environment. Deployment in a real environment as if it were a production environment and testing against this environment provide feedback on the product for further evaluation as a business idea. This chapter details the aspects related to the deployment and environment of the system and the tests carried out against it, together with the results obtained, with their subsequent evaluation.

5.1 Deployment

The project has been divided into several parts during its development favoring a development based on components. This characteristic of the project also makes it necessary to deploy each service independently in order to maintain modularity. If everything were in the same place and an error occurred, the system would stop working instantly. The following subsections detail the measures that have been taken for the correct deployment of each part, its maintenance, its deployment architecture and the measures implemented to make each service have maximum availability within the available resources.

5.1.1. Database deployment

The database is hosted on a dedicated server that can be accessed by systems running on any other instance. As mentioned above, the chosen database system is MongoDB.

One of the main advantages of using MongoDB is that you already have a system to replicate the database [1] and have multiple instances of them that are organized automatically, specifically using the Replica set capability of MongoDB, which structure can be seen in the figure 5.1. Through this it is possible to configure several MongoDB servers as a set of replicas, defining the role of each one so that the system can manage itself. This offers benefits such as high availability, database replication, and database scalability. This is a system intended to run with multiple nodes, but it is possible to isolate each of the nodes in a container and run the entire system in a single instance.

[1] Replication MongoDB manual: https://docs.mongodb.com/manual/replication/

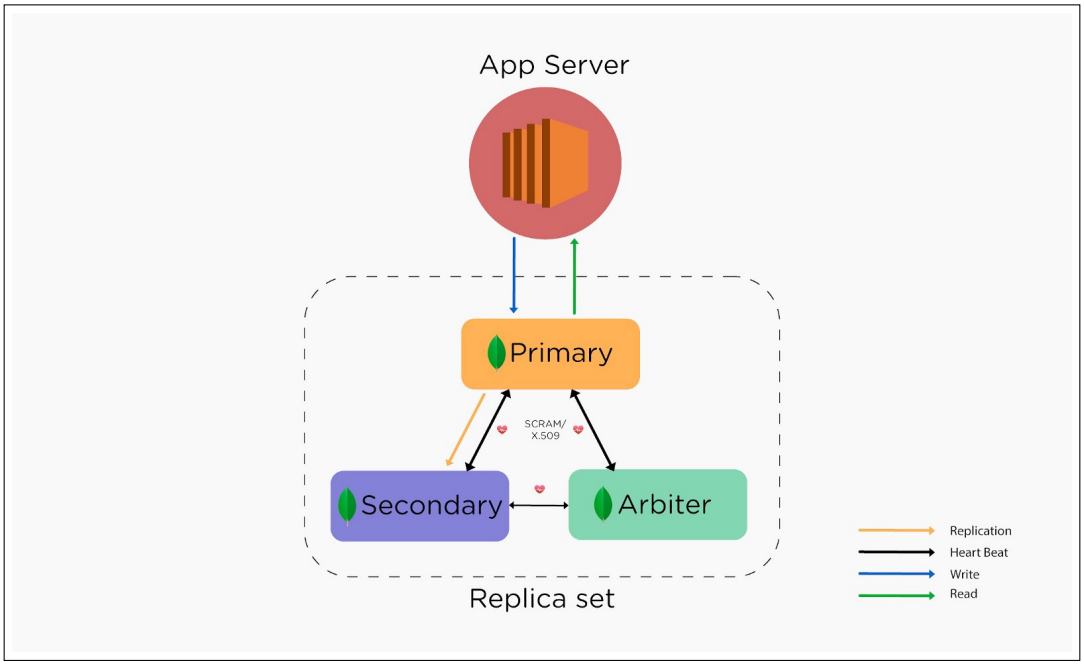


Figure 5.1: MongoDB Replica set architecture

Containerization

Thanks to using MongoDB, isolating the database manager in a container is much easier since you do not need a relational schema to start the database, it is simply necessary to run an instance of the official MongoDB image. Apart from this, there are a few things to keep in mind in order to run MongoDB in a container:

- **Mount a local volume for storage**: When a container is removed, all of its local volumes are removed along with it. To prevent this, a directory on the local machine must be mounted as the mongo data storage directory, which is located in the **/data/db** path.
- **Edit the binded IP**: By default in MongoDB only access to the manager is enabled from localhost, so access from other IP addresses must be enabled, which is done through the *–bind _ip_all* parameter.
- **Expose the port**: By default all container ports are closed to the outside, so it is necessary to bind the port we want from the container to an external port of the machine, so that it can be used from outside.

This is the necessary configuration to be able to deploy a MongoDB server without replicas. If this information is reflected in a Dockerfile, a result like the code snippet 5.1.

```yaml
version: "3.8"
services:
    mongodb:
        image : mongo
        container_name: mongodb
        environment:
            - PUID=1000
            - PGID=1000
        volumes:
            - /home/data/db:/data/db
        ports:
            - 27017:27017
        restart: unless-stopped
        entrypoint: [ "/usr/bin/mongod", "--bind_ip_all"]
```

Code Listing 5.1: MongoDB node docker-compose.yml

Using these steps it is possible to deploy an instance, but the goal is to deploy a replicated instance system using Docker. You need to create the configuration file to deploy the mirror system. This document is in charge of collecting which MongoDB instances are part of the replica system and establishing the role of each node, as can be seen in code fragment 5.2.

```
docker-compose up -d
docker exec -it localmongo1 mongo

rs.initiate(
  {
    _id : 'rs0',
    members: [
      { _id : 0, host : "mongo1:27017" },
      { _id : 1, host : "mongo2:27017" },
      { _id : 2, host : "mongo3:27017", arbiterOnly: true }
    ]
  }
)

exit
```

Code Listing 5.2: MongoDB Replica set config file docker.sh

As a second part of deploying the replica set, it is necessary to save the necessary configuration in the main node to be able to access the rest of the replicas and establish a connection. For this, a file with the configuration is created which will be added to the main node in its deployment using docker-compose, setting it as a command to be executed once the container is started. This can be seen in code fragment 5.3 on line 8.

Finally, it is necessary to configure the docker-compose with the names of the MongoDB nodes and copy the necessary configurations for its deployment. The configuration is similar to the code snippet 5.3 but with all the details of each of the nodes. In the case of this project, 1 primary node, a secondary node and an arbiterOnly node are used [2]. This provides two nodes for query processing and one node for maintaining a replica of the database only, which adds enormous redundancy to the system's data set.

[2]MongoDB Replica set Docker deployment example: https://gist.github.com/harveyconnor/518e088bad23a273cae6ba7fc4643549

```
mongo-setup:
    container_name: mongo-setup
    image: mongo
    restart: on-failure
    networks:
      default:
    volumes:
      - ./scripts:/scripts
    entrypoint: [ "/scripts/setup.sh" ] # Make sure this file
       exists (see below for the setup.sh)
    depends_on:
      - mongo1
      - mongo2
      - ...

  mongo1:
    hostname: mongo1
    container_name: localmongo1
    image: mongo
    expose:
      - 27017
    ports:
      - 27017:27017
    restart: always
    entrypoint: [ "/usr/bin/mongod", "--bind_ip_all", "--replSet"
       , "rs0", "--journal", "--dbpath", "/data/db", "--
       enableMajorityReadConcern", "false" ]
    volumes:
      - /mongo/data1/db:/data/db
      - /mongo/data1/configdb:/data/configdb
  mongo2:
  ...
```

Code Listing 5.3: MongoDB node docker-compose.yml

5.1.2. Vertical jump detector deployment

This part of the project has been treated as a separate component throughout the project, which is why it must be deployed autonomously from the rest of the system components. The main reason to isolate it is due to its processing time. Processing a video frame by frame is not something trivial or fast, so if this functionality were implemented in conjunction with the rest of the backend, when deploying everything together, the loading times would be just as slow for the entire system, no matter how many replicas were deployed.

Another point is that if this part is the one that consumes the most resources, and replica escalation is established according to consumption, for example, as many login as video processors would be deployed, when login is a quick and simple process that does not need the same level of scalability as this part of the system.

Deployment diagram

As can be seen in figure 5.2, for this service it has been decided to deploy 5 replicas in an AWS EC2 instance, all of them managed by a load balancer which will expose its interface to access the vertijump-detector service. All services are isolated in a container offering the same capacities each.

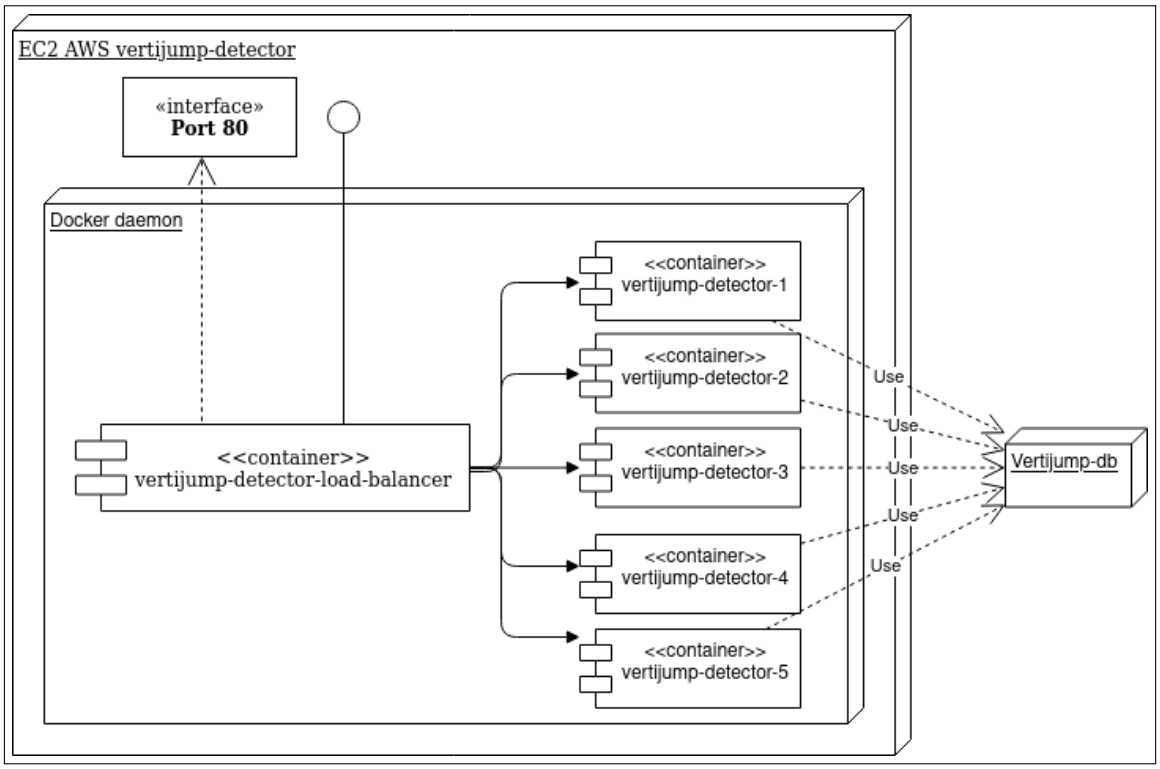


Figure 5.2: Vertijump detector deployment diagram

Containerization

In order to deploy the containers with the service, it is necessary to create the respective Dockerfiles for both vertijump-detector and its load balancer. As can be seen in the code fragment 5.4, in order to isolate this component it is necessary to use a custom Docker image with Python3 and OpenCV installed by default. This is a necessary aspect since this is the image itself that has been in charge of installing all the dependencies of the OpenCV library, and you only have to worry about the rest of the libraries through the component's *requirements.txt* file, as can be seen on the line 15 of the code snippet. Finally, it is important to note that the service is exposed on port 5000, important data for the load balancer.

After detailing how to containerize the vertijump-detector service, it is necessary to create an image for the load balancer. As can be seen in the code snippet 5.5, the image to be created takes an Nginx image as a base and adds a custom configuration file to it. This file, as can be seen in the code snippet 5.5, is in charge of filtering the traffic that receives through port 80 to port 5000 of one of the servers that it has added as a load balancer.

```
# Base image
FROM jjanzic/docker-python3-opencv

# Metadata information
LABEL maintainer="ismaelprzm@gmail.com"
LABEL version="1.0"

# Create working directory
RUN mkdir -p /opt/app

# Stablish working directory
WORKDIR /opt/app

# Install requeriments dependencies
COPY requirements.txt .
RUN . /opt/venv/bin/activate && pip install -r requirements.txt

# Copy application
COPY . .

# Install app
RUN pip install -e .

# Expose app port
EXPOSE 5000

# Init app when init server
CMD python app.py
```

Code Listing 5.4: Vertijump detector Dockerfile

```
# using Nginx base image
FROM nginx

# delete nginx default .conf file
RUN rm /etc/nginx/conf.d/default.conf

# add the .conf file we have created
COPY nginx.conf /etc/nginx/nginx.conf
```

Code Listing 5.5: Vertijump detector load balancer Dockerfile

These images are stored in the official Docker registry, DockerHub. This registry allows that, after having created an image with the dependencies and configuration necessary to run our service, it is likely to upload the image to the registry to be able to use it elsewhere by accessing the registry.

```
events {}

# Define which servers to include in the load balancing scheme.

http {
    upstream nginx {
        server nginx;
        server server_vertijump-detector_1:5000;
        server server_vertijump-detector_2:5000;
        ...
     }
     client_max_body_size 100M;


# This server accepts all traffic to port 80 and passes it to the upstream.

     server {
         listen 80;
         server_name nginx.com;
         location / {
              proxy_pass http://nginx;
          }
     }
}
```

Code Listing 5.6: Vertijump detector load balancer nginx config file

Deployment of replicas and load balancer

Once the images of the vertical jump detection service and the specific load balancer for this service are already available, it is only necessary to specify how many replicas Docker should deploy of each of the two services and put the appropriate names to the configuration file of Nginx used, as can be seen in code fragment 5.7. This allows us to access the service through a load balancer that will decide which of the 5 replicas responds to the request, usually through a Round Robin algorithm.

```
version: '3.8'
services:
    vertijump-detector:
        image: ismpere/vertijump-detector
        deploy:
            replicas: 5
        ports:
            - "5000"
        volumes:
            - /opt/app
    nginx:
        image: ismpere/vertijump-detector-load-balancer
        container_name: nginx
        ports:
            - 80:80
        depends_on:
            - vertijump-detector
```

Code Listing 5.7: Vertijump detector docker-compose file

5.1.3. Backend deployment

As it is the same as in the case of the vertical jump detector of an application made with Python, it is the same configuration making the following modifications:

- **Use a Python image as a base for the new image**: In the previous case, due to the dependency on OpenCV and its installation difficulty, a more specific image was used to create the base image of the service, as can be seen in the code fragment 5.4. Unlike that case, this service only needs an official Pyhon image, so only the image field would have to be modified.
- **Decrease the number of replicas to 3**: This service consumes fewer resources than the previous one, so with 3 replicas deployed it is more than enough.
- **Edit the nginx configuration file**: To be able to create the image of the balancer as in the previous case, you only have to edit in the code snippet 5.6 the name of the services that the load balancer routes, since now it is the backend, not from vertijump-detector.

With these steps, all the previous documents are reusable for the deployment of this service in the AWS EC2 instance dedicated to hosting this service.

Deployment diagram

After applying the aforementioned modifications, the deployment diagram for this service is shown in figure 5.3. There are 3 replicas of the service deployed in independent containers and managed by a load balancer, which exposes its port 80 to access the backend service.

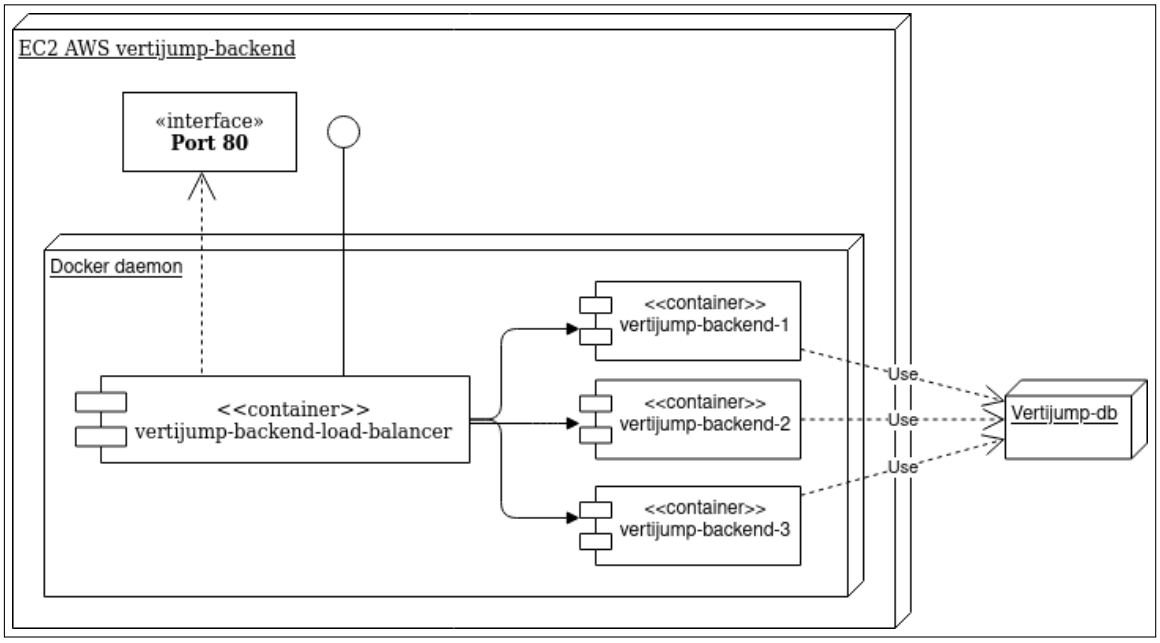


Figure 5.3: Vertijump backend deployment diagram

CHAPTER 6
Testing

This chapter details the tests carried out to verify the correct operation of the developed system. It should be noted that there are two very different parts to this chapter: the jump detector tests and the complete system tests.

6.1 Vertical jump detector

This section will expose all the tests done to check if the vertical jump detector meets the precision requirements set out in subsection 3.2.2. This part of the project has been developed from a more investigative perspective, founding a pass for the tests, establishing hypotheses and contrasting the information to verify or not those hypotheses.

Each test section has an established hypothesis to be tested, so the tests that this subsection contains will be evaluated from a specific criterion, always trying to contrast the hypothesis. It is important to rule out that it is not a research process as such, since it has not been a development as such, but a hybrid between both models to try to get the best of each one.

6.1.1. Vertical jump detection parameter tests

In this section the tests are focused on testing different settings of the parameters that have been established to detect vertical jumps.

Modifiable parameters

The following are the two types of parameters that can be modified for these detection tests:

- **Constants used in detection**: These parameters are decimal values that represent a limit or margin of error. Specifically, these values are those mentioned below:
  - **Minimum area percentage to detect movement**: This is a decimal value that represents the minimum percentage that a detected contour must occupy within the frame to be processed in order to be considered as a valid contour. The value of this parameter is based on the percentage that occupies a foot placed sideways to the camera at a distance of 1m 10cm, a big step of a 1.75m tall person.

  - **Percentage of margin of error of the baseline**: This is a decimal value that represents the percentage that includes the margin of error of the baseline, both up and down the baseline (Y axis). The value of this parameter is based on the minimum jump of an average person and the range of motion of one foot while positioning to perform a vertical jump.
  - **Minimum time in air**: This is a decimal value that represents the minimum time in the air that a person has to be to be considered a valid vertical jump. This value is based on the least effort jump that an average person can make by subtracting a margin of error.
  - **Maximum time in air**: This is a decimal value that represents the maximum air time that a jump can take to be considered valid. The value of this parameter is based on the world record for vertical jump, set by Watch Josh at 47.1'or 1.19634m [1].

- **Frame processing parameters**: These parameters refer to values used in the transformations applied to the video frames in the preprocessing. Changing these conditions can result in a very different image and a detection result unrelated to those tested with other settings. Specifically, the following values can be modified:

  - **Gaussian blur**: This is a transformation commonly used to soften an image and reduce image noise. In turn, two parameters can be modified from this transformation:
    * Kernel: It is the kernel used in each of the blurring of the image.
    * Iterations: Number of iterations of the blur with the specified kernel.
  - **Threshold**: It is a transformation that creates an image with two bits of color from a certain threshold. The modifiable parameters of this transformation are:
    * Threshold: Value from which the division of the color bits is made.
    * Maximum value: Maximum color value to be set if the threshold is exceeded.
    * Threshold type.
  - **Dilate**: This transformation stretches the edges of the image outlines. It is a transformation commonly used to smooth outlines and fill any gaps in open outlines. The modifiable values of this transformation are:
    * Kernel: Kernel used in each of the image dilatations.
    * Iterations: Number of iterations of dilation to perform with that kernel.

Test resources

For these tests there are 45 videos of 2 seconds duration in which a vertical jump is made. In these tests, 11 different subjects have been included, some of them jumping in up to 5 different positions and at 3 different distances.

Hypothesis

Test case 1 will present the best results as it is using the base values calculated for the detector.

[1] Vertical jump world record video: https://www.youtube.com/watch?v=lgkCxnSHV7w

Test cases

The test configurations chosen for these tests are detailed below:

Test case ID	1	2	2
Min frame area Percentage to detect movement	0.002	0.00175	0.002
Margin error percentage baseline	0.025	0.0225	0.025
Min jump time in air	0.25	0.25	0.25
Max jump time in air	1.25	1.25	1.25
Gaussian blur kernel	7x7	7x7	7x7
Gaussian blur iterations	5	5	7
Threshold value	20	20	30
Threshold maximum value	255	255	255
Threshold type	BINARY	BINARY	BINARY
Dilate kernel	None	None	None
Dilate iterations	0	0	2

Table 6.1: Test cases of parameter tests

Results

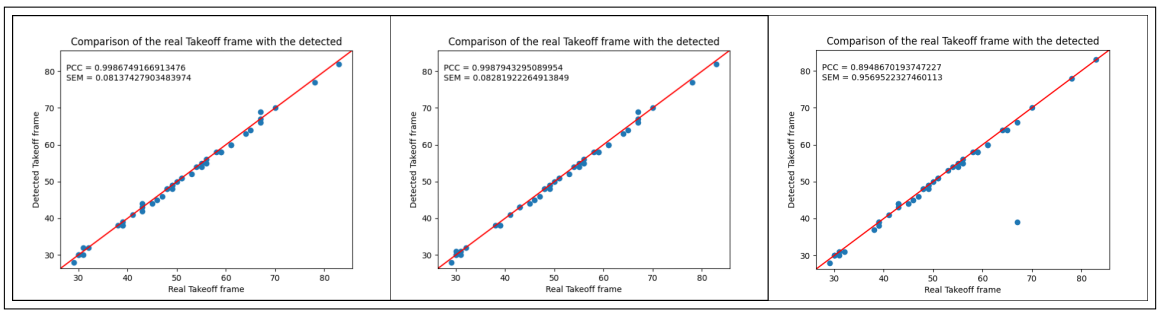


Figure 6.1: Correlation graph of detected takeoff frame with real - Parameter tests

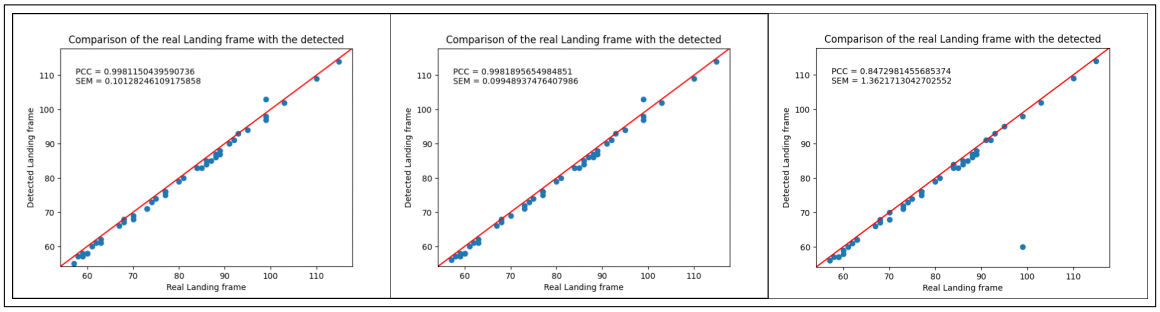


Figure 6.2: Correlation graph of detected landing frame with real - Parameter tests

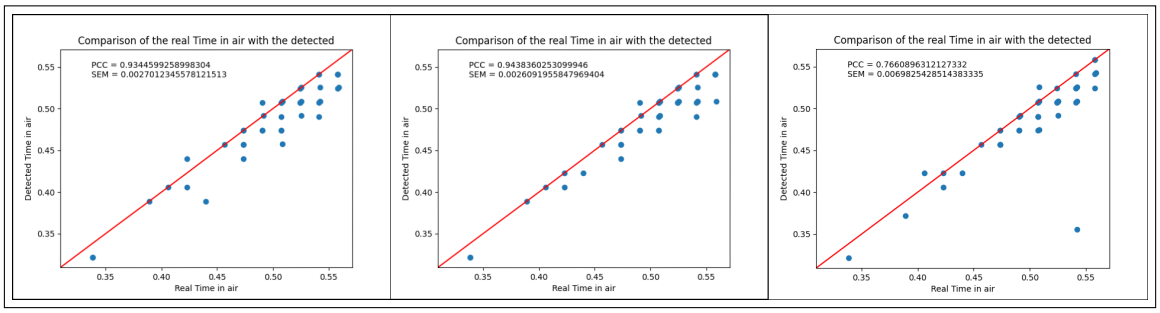


Figure 6.3: Correlation graph of detected time with real - Parameter tests

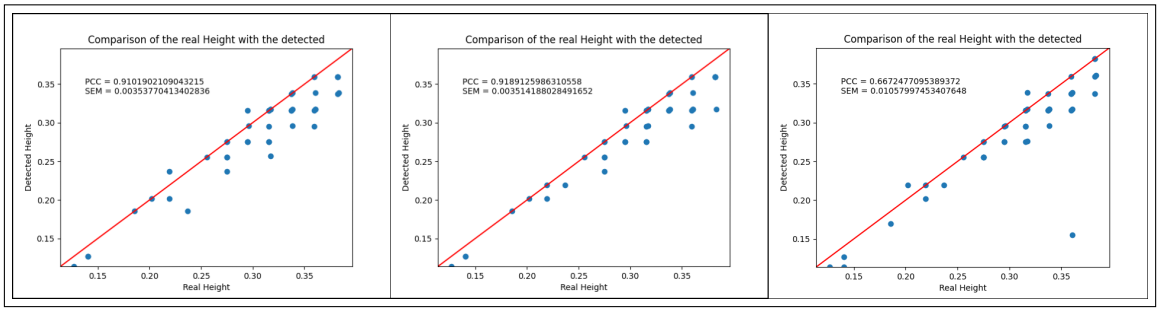


Figure 6.4: Correlation graph of detected height with real - Parameter tests

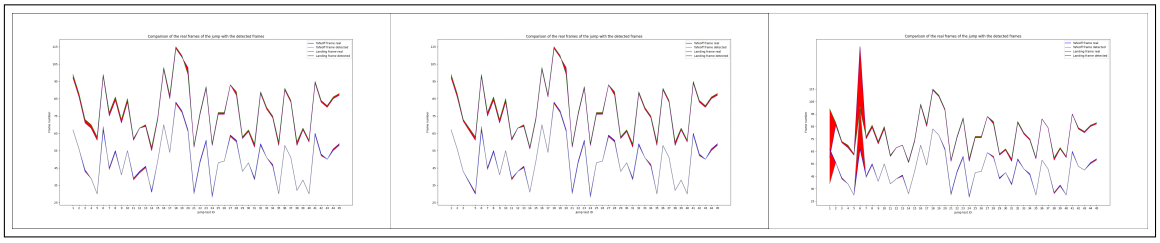


Figure 6.5: Comparison of detected jump frames with real - Parameter tests

Parameter	Tafeoff frame error	Landing frame error	Time in air error	Height error
Mean	0.555	1.355	0.0188	0.0229
Median	1	1	0.0169	0.0199
Mode	1	1	0	0
Standard desviation	0.5398	0.6718	0.0179	0.0235
Variance	0.2914	0.4514	0.0003	0.0005
Min	0	0	0	0
Max	2	4	0.0846	0.1210
Amplitude	2	4	0.0846	0.1210
Standard mean error	0.0814	0.1013	0.0027	0.0035
Q 25%	0	1	0	0
Q 50%	1	1	0.0169	0.0199
Q 75%	1	2	0.0338	0.0407
Failed tests	0 / 45			

Table 6.2: Descriptive values of the errors of test case 1 - Parameter tests

Parameter	Tafeoff frame error	Landing frame error	Time in air error	Height error
Mean	0.5227	1.2727	0.0165	0.0203
Median	0.5	1	0.0169	0.0196
Mode	0	1	0	0
Standard desviation	0.5430	0.6523	0.0171	0.0230
Variance	0.2949	0.4261	0.0003	0.0005
Min	0	0	0	0
Max	2	4	0.0846	0.1210
Amplitude	2	4	0.0846	0.1210
Standard mean error	0.0828	0.0994	0.0026	0.0035
Q 25%	0	1	0	0
Q 50%	0.5	1	0.0169	0.0196
Q 75%	1	2	0.0169	0.0228
Failed tests	1 / 45			

Table 6.3: Descriptive values of the errors of test case 2 - Parameter tests

Parameter	Tafeoff frame error	Landing frame error	Time in air error	Height error
Mean	1.8	3.0444	0.024	0.0309
Median	0.5	1	0.0169	0.0199
Mode	0	1	0	0
Standard desviation	6.3477	9.0356	0.0463	0.0702
Variance	40.2933	81.6424	0.0021	0.0005
Min	0	0	0	0
Max	34	50	0.2706	0.4491
Amplitude	34	50	0.2706	0.4491
Standard mean error	0.9569	1.3622	0.0069	0.0106
Q 25%	0	1	0	0
Q 50%	0	1	0.0169	0.0199
Q 75%	1	2	0.0169	0.0222
Failed tests	0 / 45			

Table 6.4: Descriptive values of the errors of test case 3 - Parameter tests

Analysis

Discussing about the detection of the takeoff frame, as we can see in the figure 6.1, the test case that achieves a greater correlation between the real data and those obtained is the case of test 2, although the Pearson coefficient of the test case 1 is almost the same as in test case 2. If the data in tables 6.2, 6.3 and 6.4 are also observed, it can be observed that the lowest mean error in the detection of the take-off frame is as in the previous case the case of test 2. An aspect that is also remarkable is the awfull detection error of test case 3 that can be observed in the figure 6.5.

Regarding the detection of the landing frame, observing figure 6.2, something similar happens as with the take-off frame, but this time with a much smaller difference in the Pearson coefficient. The test case that achieves a greater correlation between the real data and those obtained is the test case 2. If the data in tables 6.2, 6.3 and 6.4 are observed again, it is easy to see that the lower mean error in the detection of the landing frame is as in the previous case the test case number 2, this time with a greater margin of gain than in the previous case. A remarkable aspect again is the awful detection error of test case 3 that can be observed in figure 6.5.

Analyzing figure 6.3, as expected by the results of the two previous variables, if the time in the air detected is analyzed, the test case with the highest correlation coefficient is the case of test 2. If we also look at the tables 6.2, 6.3 and 6.4 are also observed as is the case of the test with the lowest mean error of all. Finally, again it is seen that the mean error of test case 3 is considerably worse than the other two test cases.

Finally, analyzing the variable height reached by means of figure 6.4, and as a consequence of the other 3 variables, test case 2 is once again the one with the highest percentage of correlation. If tables 6.2, 6.3 and 6.4 are analyzed, we obtain as data that test case 2 presents the lowest mean error for this variable.

A relevant aspect is that the option that provides the best results, the case of test 2, is the only one that has failed in the detection of a jump, since for example in the case of

test 3 it can be observed that there is a data that comes out of the average but has come to detect something.

Conclusion

Test case 2 provides a slight improvement in the percentage of detected error, but it is the only algorithm that has failed in detection, so in this case test case 1 is chosen as the best option by prioritizing robustness by above precision. The hypothesis is fulfilled.

6.1.2. Vertical jump distances tests

In this section the tests are focused on evaluating the accuracy of the vertical jump detection by performing the jump at several different distances from the camera.

Test cases setup

To carry out the tests at different distances, several different jump points have been established at certain measurements. Both the position of the camera and the different jump points are marked on the ground as can be seen in figure 6.6. This setup has been made to record a total of 3 jumps, one at each distance, to be able to evaluate from what distance vertical jump is better detected. These distances have been chosen taking into account the measurement of a step of a person of average height, around 1.70m tall. As for the camera, it is a mobile device located on a stand to keep the phone vertically recording.

Figure 6.6: Distance tests setup.

After the results of the first tests, the configuration of the detection and processing parameters are those established by test case number one, the test case selected as the best detection option.

Test resources

To carry out these tests, a total of 11 vertical jumps are available for each of the 3 distances to be tested, which gives a total of 33 tests.

Hypothesis

Test case number 2 has the highest detection, since it is the minimum distance in which the feet are seen during the video plus a margin of error, and it is the reference distance for the other two tests.

Test cases

The test cases in this case are very easy to define, they are the 3 distances marked on the ground with three blue X, as can be seen in the figure 6.6. One aspect to keep in mind is that when performing the tests, users should try to land as close to the starting position as possible, in order to assess the distance more specifically. They are specifically located at the following distances:

- **Test case 1**: 110cm
- **Test case 2**: 120cm
- **Test case 3**: 130cm

Results

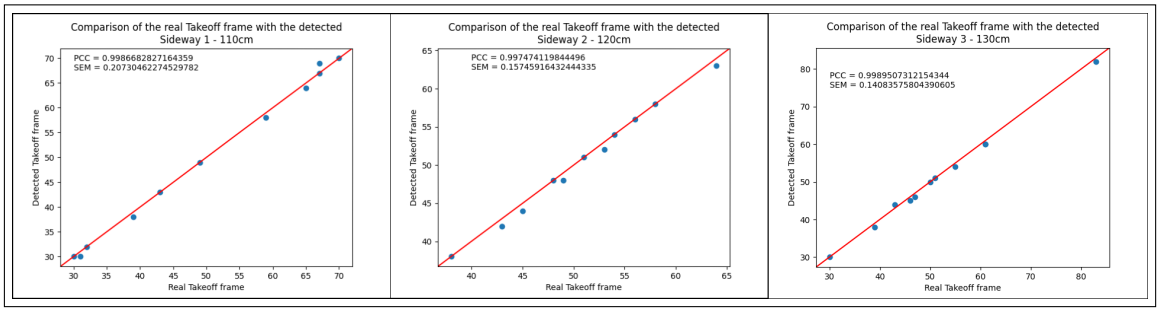


Figure 6.7: Correlation graph of detected takeoff frame with real - Distance tests

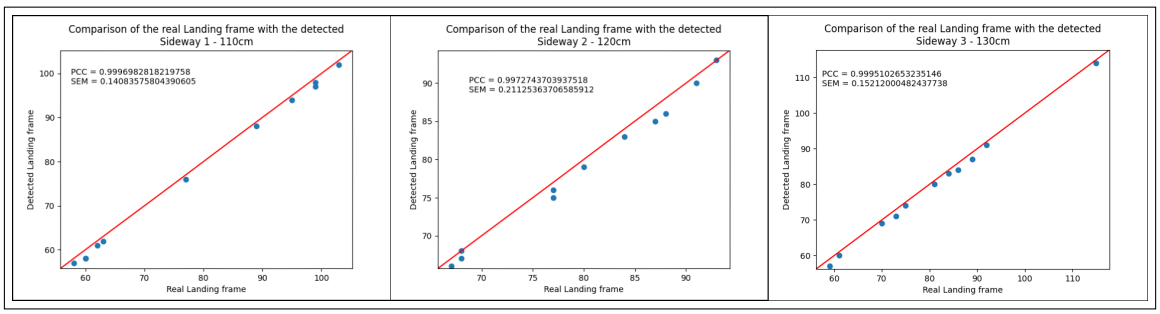


Figure 6.8: Correlation graph of detected landing frame with real - Distance tests

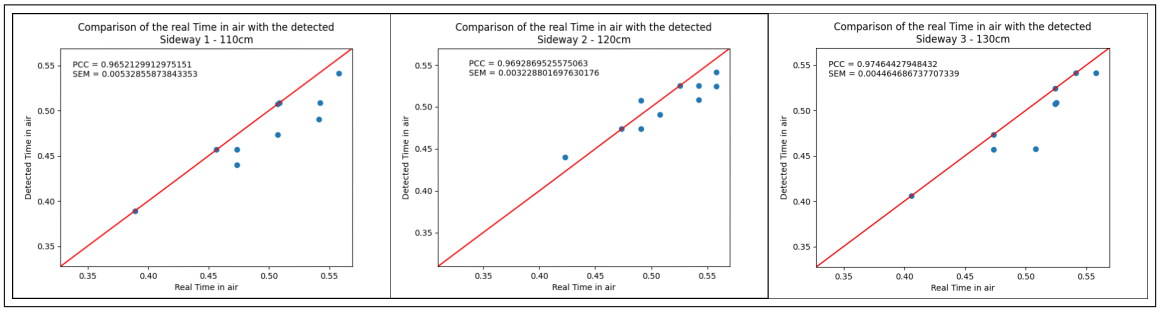


Figure 6.9: Correlation graph of detected time with real - Distance tests

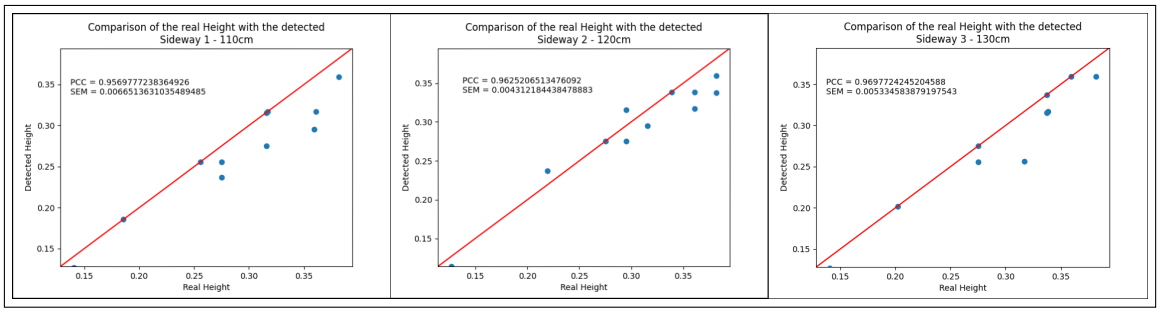


Figure 6.10: Correlation graph of detected height with real - Distance tests

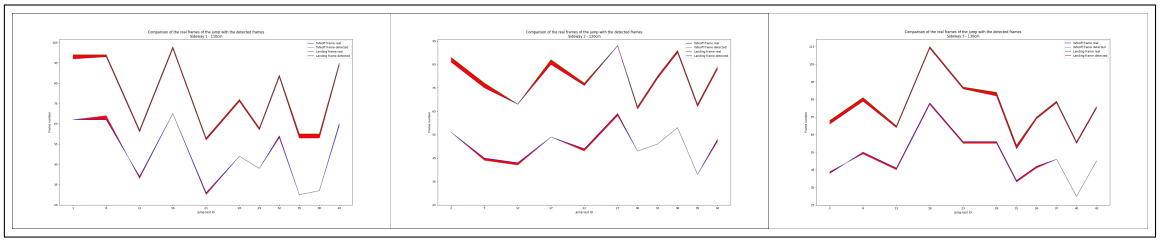


Figure 6.11: Comparison of detected jump frames with real - Distance tests

Analysis

Regarding the analysis on the detection of the takeoff frame, it can be seen in figure 6.5 that the Pearson correlation coefficient is higher in the case test 3. This information is contrasted with the 6.5, 6.6 and 6.7 tables. The tables are the opposite of what would be expected, since the test case 3 has the worst mean error, the best being that of the test case 2. The correlation of the test case 3 agrees with its standard error of the mean, but the test case 2 has a very similar standard error of the mean, so in this case it is the test case 2 that obtains the best results overall.

Regarding the detection of the landing frame, after viewing figure 6.6, it is extracted that the test case 1 is the one with the best correlation, but by very little compared to the test case 3. After this, observing the tables mentioned, it is concluded that the mean error of the test case 2 is the best by far, although it presents a fairly high variance compared to the other case tests. In this case, the test case 2 is the one with the best result.

Treating the airtime error, if you look at figure 6.9, you can see that the test case 3 is the one with the best correlation, something that did not seem very intuitive after the data

Parameter	Tafeoff frame error	Landing frame error	Time in air error	Height error
Mean	0.5454	1.2727	0.0185	0.0220
Median	0.0	1	0.0169	0.0192
Mode	0	1	0	0
Standard desviation	0.6555	0.4454	0.0168	0.0210
Variance	0.4297	0.1983	0.0003	0.0004
Min	0	1	0	0
Max	2	2	0.0507	0.0642
Amplitude	2	1	0.0507	0.0642
Standard mean error	0.2073	0.1408	0.0053	0.0066
Q 25%	0	1	0	0
Q 50%	0	1	0.0169	0.0193
Q 75%	1	1.5	0.0338	0.0393
Failed tests	0 / 11			

Table 6.5: Descriptive values of the errors of test case 1 - Distance tests

Parameter	Tafeoff frame error	Landing frame error	Time in air error	Height error
Mean	0.4545	1.0909	0.0169	0.0205
Median	0.0	1	0.0169	0.0207
Mode	0	1	0	0
Standard desviation	0.4979	0.6680	0.0102	0.0136
Variance	0.2479	0.4462	0.0001	0.0001
Min	0	0	0	0
Max	1	2	0.0339	0.0449
Amplitude	1	2	0.0339	0.0449
Standard mean error	0.1574	0.2112	0.0032	0.0043
Q 25%	0	1	0.0169	0.0154
Q 50%	0	1	0.0169	0.0206
Q 75%	1	1.5	0.0169	0.0224
Failed tests	0 / 11			

Table 6.6: Descriptive values of the errors of test case 2 - Distance tests

Parameter	Tafeoff frame error	Landing frame error	Time in air error	Height error
Mean	0.7272	1.3636	0.0138	0.0164
Median	1	1	0.0169	0.0193
Mode	1	1	0	0
Standard desviation	0.4453	0.4810	0.0141	0.0169
Variance	0.1983	0.2314	0.0002	0.00028
Min	0	1	0	0
Max	1	2	0.0508	0.0602
Amplitude	1	1	0.0508	0.0602
Standard mean error	0.1408	0.1521	0.0045	0.0053
Q 25%	0.5	1	0	0
Q 50%	1	1	0.0169	0.0193
Q 75%	1	2	0.0169	0.0214
Failed tests	0 / 11			

Table 6.7: Descriptive values of the errors of test case 3 - Distance and Position tests

from the previous arguments. Adding the information present in the tables mentioned, it is observed that the test case 3 has a much better mean error despite having a slightly higher variance than the test case 2. In this case, it is the test case 3 that presents better results.

Treating the data presented in figure 6.10 about the error of the height reached, the test case 3 has a higher correlation index. Finally, regarding the data presented in figure 6.11, it does not clarify much which of the three test cases has a better performance, since at first glance it seems that it is the first but we have verified with its descriptive values that it does not have the best results. If the aforementioned tables are contrasted, the mean error is much lower, so test case 3 is the one with the best performance in detecting the jump height.

Conclusion

The case test 3 is the one that offers the best result. This may be due to the fact that being a little further away the contours to be drawn are clearer and it does not detect as much movement. The hypothesis is not fulfilled.

6.1.3. Position tests

In this section, the tests focus on checking in which position the user should take to better detect a vertical jump.

6.1.4. Test cases setup

As can be seen in figure 6.12, to carry out these tests, three different positions have been established in which the user must position himself to perform the vertical jumps:

- Facing the camera

- With the back to the camera
- Putting the camera aside, no matter which

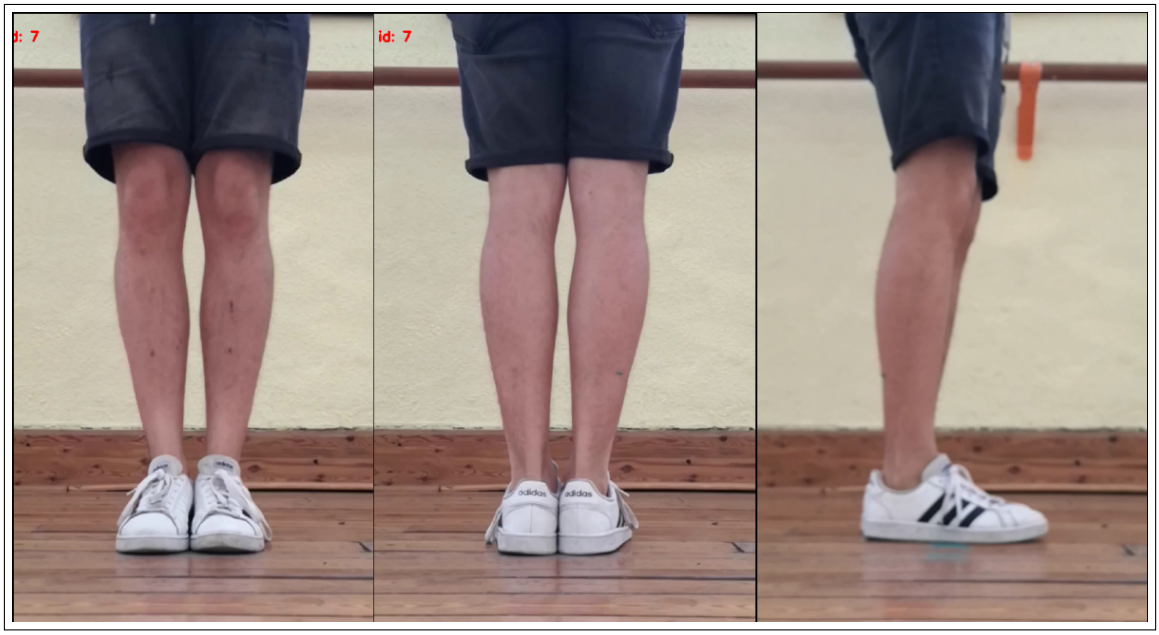


Figure 6.12: Positions for position tests

Test resources

To carry out these tests, a total of 6 jumps are available for the first case test, 6 jumps for the second and 11 jumps for the third, which makes a total of 23 vertical jumps available to obtain the results.

Hypothesis

The test case number 3, performing the vertical jump leaving aside the camera, is the one that should provide the best result feet take off and landing is better visualized and there is more contour to detect.

Results

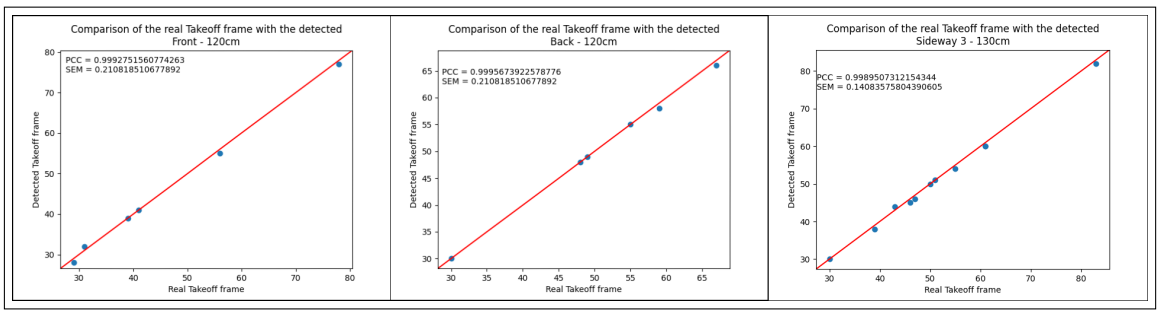


Figure 6.13: Correlation graph of detected takeoff frame with real - Distance tests

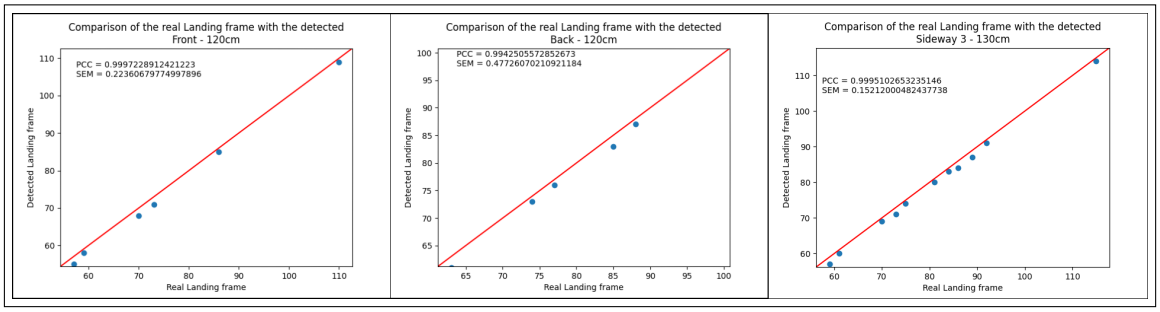


Figure 6.14: Correlation graph of detected landing frame with real - Distance tests

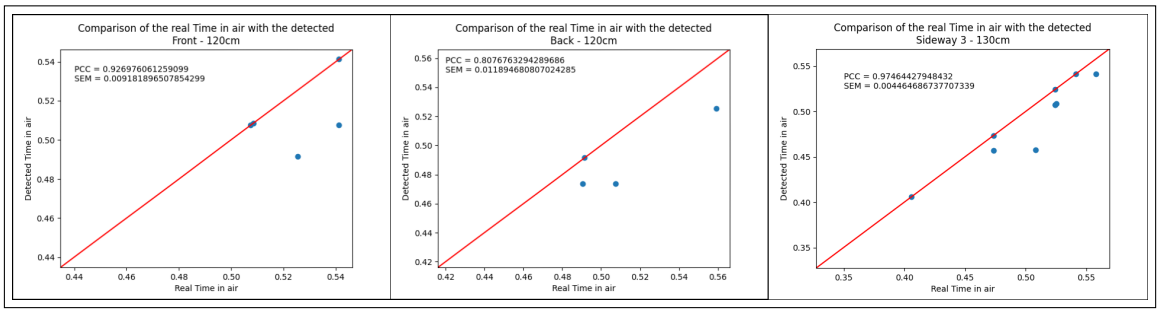


Figure 6.15: Correlation graph of detected time with real - Distance tests

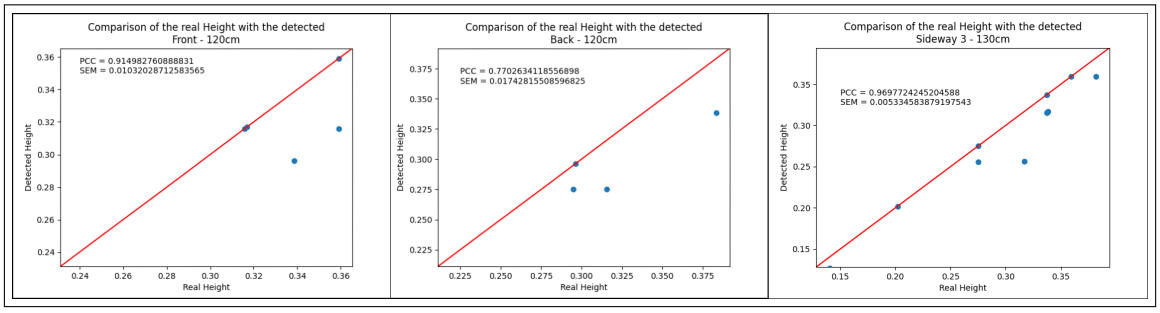


Figure 6.16: Correlation graph of detected height with real - Distance tests

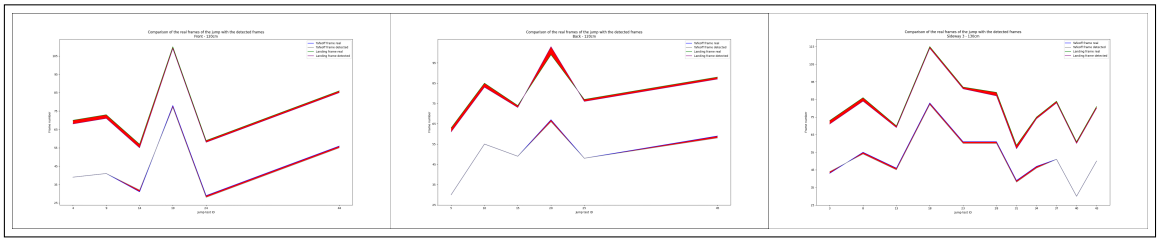


Figure 6.17: Comparison of detected jump frames with real - Distance tests

Parameter	Tafeoff frame error	Landing frame error	Time in air error	Height error
Mean	0.6666	1.5	0.0197	0.0229
Median	1	1.5	0.0169	0.0211
Mode	1	1	0	0
Standard desviation	0.4714	0.5	0.0205	0.0230
Variance	0.2222	0.25	0.0004	0.0005
Min	0	1	0	0
Max	1	2	0.0507	0.0516
Amplitude	1	1	0.0507	0.0516
Standard mean error	0.2108	0.2236	0.0091	0.0103
Q 25%	0.5	1	0	0
Q 50%	1	1.5	0.0169	0.0211
Q 75%	1	2	0.0338	0.0429
Failed tests	0 / 6			

Table 6.8: Descriptive values of the errors of test case 1 - Position tests

Parameter	Tafeoff frame error	Landing frame error	Time in air error	Height error
Mean	0.3333	1.8333	0.0310	0.0407
Median	0	1.5	0.0253	0.0303
Mode	0	1	0	0
Standard desviation	0.4714	1.0671	0.0266	0.0389
Variance	0.2222	1.1388	0.0007	0.0015
Min	0	1	0	0
Max	1	4	0.0846	0.1210
Amplitude	1	3	0.0846	0.1210
Standard mean error	0.2108	0.4773	0.0119	0.0174
Q 25%	0	1	0.0169	0.0185
Q 50%	0	1.5	0.0254	0.0303
Q 75%	0.5	2	0.0338	0.0423
Failed tests	0 / 6			

Table 6.9: Descriptive values of the errors of test case 2 - Position tests

Analysis

Regarding the take-off frame, observing figure 6.13 and tables 6.8, 6.9 and table 6.7 of the previous tests, since they are the same results, it is observed that the 3 test cases have a very good correlation coefficient, but case test two obtains the best mean error with a difference to the rest of the test cases. On the part of the landing frame, with the help of figure 6.14, the high correlation coefficient of the 3 case tests is checked again, but it is the third that presents a better mean error in detection by far.

About the time in the air, as shown in the figure 6.15 the first two tests have very little correlation, but on the other hand, the mean of the second is not bad, but finally the third test case has the best mean error of the time in air. Finally, after viewing figure 6.16, a case similar to air time is seen, with little correlation in the first two test cases but higher in the third and with a better mean error. This makes the case 3 test the best in all respects. Figure 6.17 has not been very useful on this occasion, since having few tests in the first two tests, the graph is a bit poor and without much to highlight against the third.

Conclusion

Despite presenting some good detection results by jumping from the front or back to the camera, having a larger surface area to detect when facing the camera makes detection more accurate in this way. The hypothesis has been fulfilled.

CHAPTER 7
Conclusions

In this work, an application consisting of a vertical jump detection system has been developed, reaching the detection and precision objectives. I have learned to integrate totally different systems with each other with technologies that sometimes require more than technological knowledge. This has led to a much more in-depth study of the design, architectures, and communication patterns such as REST APIs.

I have learned to implement a real system using systems prepared for production environments, facing problems such as in which country to host an application or how much redundancy to maintain, which have been critical aspects of the system. I have also learned how to containerize applications to deploy them in a scalable way using docker-compose and AWS, using that knowledge in system deployment.

A small-scale investigation has been completed while its result has been integrated with a real system. This has contributed to improve my knowledge in the processes of documentation, testing and validation of a hypothesis, the possibility of detecting a vertical jump in a video. Through tests to validate this hypothesis, it has been obtained that the best position to detect a vertical jump is to perform the jump standing sideways to the camera at a distance of 120cm. Developing this work I have acquired knowledge of computer vision, a field that has often had more to do with mathematical transformations than with a library function, which opens up more job opportunities.

I have expanded my knowledge of Android, developing a complete application that connects to a server via REST API and interacts with the data and the user. For this reason, I have learned tools such as video processing through FFmpeg or libraries for REST API requests such as Retrofit2. Finally, due to this point I have acquired a wide knowledge base about Kotlin, learning to use it for Android programming, taking all its advantages over Java.

Having a multi-component system, it has been a problem to be able to switch from one technology to another at the beginning. This has helped me outperform myself and I have improved my ability to adapt to new development technologies much faster than before. Since this project began at a university in Poland and ended in Valencia, I have found that the approach to a work of this type is totally different depending on the institution or the country, a fact that improves my ability to adapt to change, both in terms of both cultural and technological.

Finally, due to the dimensions of this work, I have had to learn to manage my work by myself, which was a problem at first, but now I know my limits and I know how to manage and organize my time much better. Due to these problems I want to continue developing in the field of project management.

7.1 Relationship of the work carried out with the studies completed

Aspects of the studies completed that have been relevant to carry out this project are detailed below:

- **IT Governance, Management and IT Project Planning and Management**: The knowledge acquired in these subjects has been the basis for managing a development as long as a master's thesis.
- **Configuration and Optimisation of Computing Systems**: All the scalable deployment part and maintaining high availability is thanks to this subject in which I was taught a very firm base of these technologies and work philosophy.
- **Computer Networks and Security**: Aspects learned in this subject, such as port and subnet management, have been key to solving problems that arose in development, such as accessing remote elements in an authenticated way with a private network.
- **Audit, Quality and Management of Information Systems**: The bases on design patterns and always betting on a clean and orderly code has made development much easier and more legible over time.
- **High-Performance Computing**: Image processing is something that requires a lot of resources and being able to have learned a base of parallelism has allowed me to experiment with tools such as TensorFlow and thus be able to assess more options for development.
- **Distributed Applications and Services**: This course has laid the foundations of my knowledge in distributed applications and component-based development, knowledge which is applied at work by dividing the detector into a scalable component.

Bibliography

[1] Department General of Statistics and Studies - Ministry of Education, Culture and Sport of Spain. Survey of sporting habits in Spain 2015 - Synthesis of results. https://www.culturaydeporte.gob.es/dam/jcr:eb913bf1-a897-403b-b056-49b42582ff37/survey-of-sporting-habits-in-spain-2015-synthesis-of-results.pdf. February, 2016. Last access on 15/05/2021.

[2] Department General of Statistics and Studies - Ministry of Education, Culture and Sport of Spain. Survey of sporting habits in Spain 2020 - Synthesis of results. https://www.culturaydeporte.gob.es/dam/jcr:56643289-95f4-4242-891d-859815f84c9d/encuesta-de-habitos-deportivos-2020-sintesis-de-resultados.pdf. June, 2021. Last access on 15/05/2021.

[3] Darmiento, A., Galpin, A.J., Brown, L.E. Vertical Jump and Power. *Strength and Conditioning Journal*, 34(6):34-42 doi: 10.1519/SSC.0b013e3182752b25, December, 2012.

[4] López-Segovia, M., Marques, M.C., Van den Tillaar, R., González-Badillo, J.J. Relationships Between Vertical Jump and Full Squat Power Outputs With Sprint Times in U21 Soccer Players. *Journal of Human Kinetics*, 30:135-144 doi: 10.2478/v10078-011-0081-2, December, 2011.

[5] Moir, G.L. Three Different Methods of Calculating Vertical Jump Height from Force Platform Data in Men and Women. *Measurement in Physical Education and Exercise Science*, 12(4):207-218 doi: 10.1080/10913670802349766, October, 2008.

[6] Espert Bosch, M.C. Estimación de la altura en el test de salto vertical mediante técnicas de procesado de sonido. TFG. Universidad Politécnica de Valencia. https://riunet.upv.es/bitstream/handle/10251/124623/Espert%20-%20Estimaci%C3%B3n%20de%20la%20altura%20en%20el%20test%20de%20salto%20vertical%20mediante%20t%C3%A9cnicas%20de%20procesado%20de%20s....pdf?sequence=1. February, 2019. Last access on 25/05/2021.

[7] Montalvo, S., Dorgo, S., Tune, C., Sapien, C., Gonzalez, M., Sanchez, J. Validity of Vertical Jump Measuring Devices. *International Journal of Exercise Science*, 2(10):69, February, 2018.

[8] Cabarkapa, D., Fry, A.C., Hermes, M.J. Accuracy of an Experimental Accelerometer for Assessing Countermovement Vertical Jump Height. *Sports Innovation Journal*, 2:45–55 doi: 10.18060/24831, 2021.

[9] Bogataj, Š., Pajek, M., Andrašić, S., Trajković, N. Concurrent Validity and Reliability of My Jump 2 App for Measuring Vertical Jump Height in Recreationally Active Adults. *Sports Performance and Health*, 10(11):3805 doi: 10.3390/app10113805, 2020.

[10] Balsalobre-Fernández, C., Glaister, M., Lockey, R.A. The validity and reliability of an iPhone app for measuring vertical jump performance. *Measurement in Physical Education and Exercise Science*, 33(15):1574-1579 doi: 10.1080/02640414.2014.996184, January, 2015.

[11] Goguen, J.A., Linde, C. Techniques for requirements elicitation. *Proceedings of the IEEE International Symposium on Requirements Engineering*, 152-164 doi: 10.1109/ISRE.1993.324822, 1993.

[12] Sayers, S.P., Harackiewicz, D.V., Harman, E.A., Frykman, P. N., Rosenstein, M.T. Cross-validation of three jump power equations. *Medicine & Science in Sports & Exercise*, 31(4):572-577 doi: 10.1097/00005768-199904000-00013, April, 1999.

[13] Johnson, D.L., Bahamonde, R. Power Output Estimate in University Athletes. *Journal of Strength and Conditioning Research*, 10(3):161-166 doi: 10.1.1.599.753, 1996.